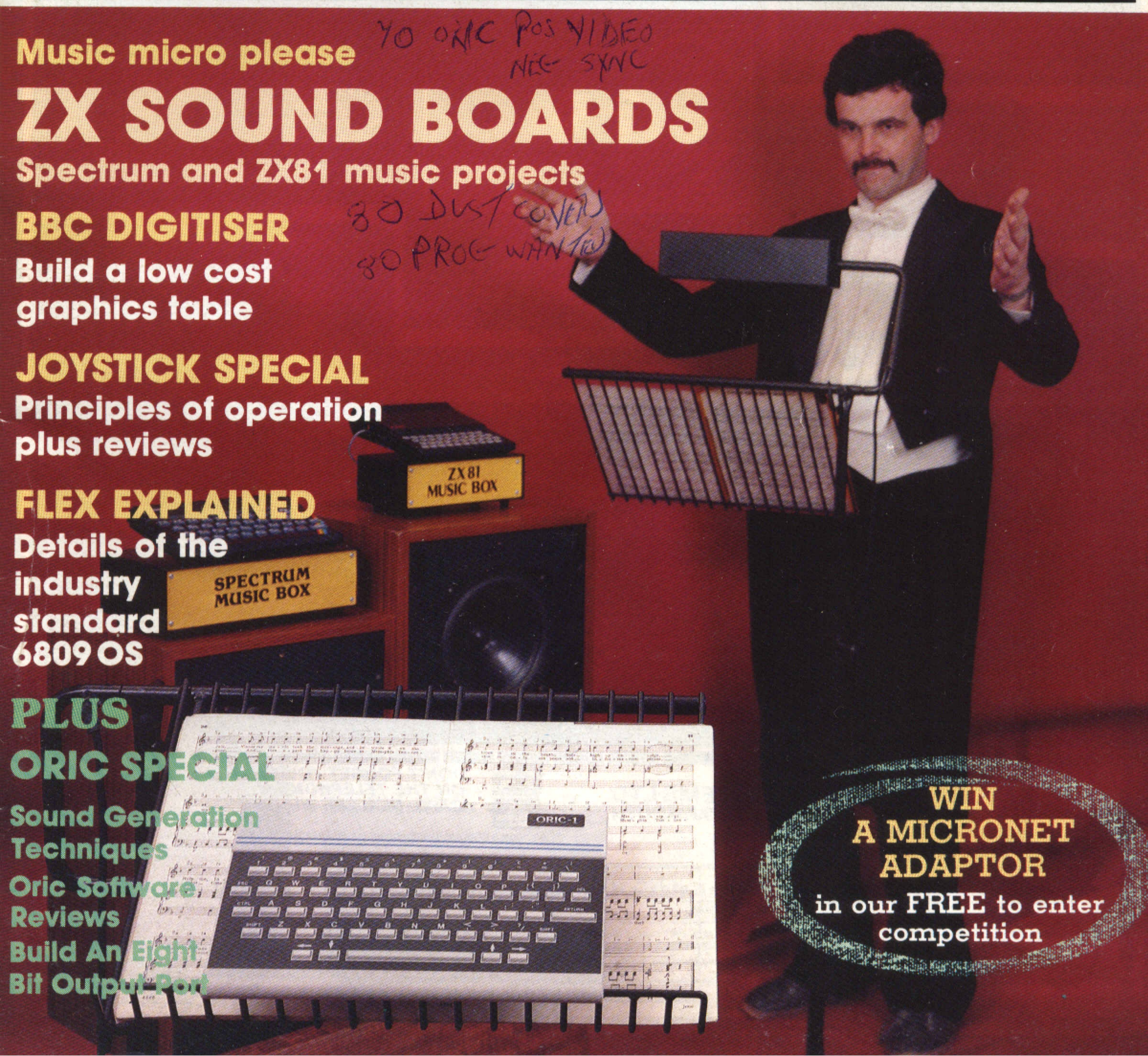
August 1983

AN EMAP PUBLICATION **75p**

Electronics & Computing MONTHLY

Britain's **First** Electronics & Computer **Applications** Magazine

9 ORIC 14 SOUND 21 ASSEMBLER 26 ORIC O/P PORT
29 VIEW WORD PROC 35 OXO [illegible]
ONL [illegible]
39 GRAPH TABLE
43 [illegible]
44
46 [illegible]
50
52 [illegible]
57 [illegible]
59 [illegible]
60 PROG TECH
62 MICRO PROC INFO 67 ERRORS 69 ORIC INFO 70 ORIC TAPES ETC 75 IC DATA 77 MODEM KIT 78 ORIC TAPE 79 ORIC TAPES COVERED



Music micro please

ZX SOUND BOARDS

Spectrum and ZX81 music projects

70 ORIC POS VIDEO NEG SYNC

80 DUST COVERS
80 PROG WANTED

BBC DIGITISER
Build a low cost graphics table

JOYSTICK SPECIAL
Principles of operation plus reviews

FLEX EXPLAINED
Details of the industry standard 6809 OS

PLUS
ORIC SPECIAL
Sound Generation Techniques
Oric Software Reviews
Build An Eight Bit Output Port

WIN A MICRONET ADAPTOR
in our FREE to enter competition

Electronics & Computing Monthly
Scriptor Court,
155 Farringdon Road,
London,
EC1R 3AD

EDITORIAL
01-833-0846
Editor
Gary Evans

Editorial Assistant
Ann Houghton

Administration
Liz Gregory

ADVERTISING
01-833-0531
Manager
Claire Fullerton

Deputy Manager
Alun Evans

Classified
Nik Saha

PRODUCTION
01-833-0846
Design
Pat Haylock
Terry Gibbins

Make-up
Time Graphics

Publisher
David Raven

Distribution
EMAP
National Publications

Published By
EMAP Business And Computer Publications

Printed by
EMLP Peterborough

Subscriptions
Electronics & Computing Monthly, (Subscription Department), Competition House, Farmdon Road, Market Harborough, Leicestershire.

ABC
MEMBER OF THE AUDIT BUREAU OF CIRCULATION

Vol 3

IN THIS ISSUE

Issue 8

See Page 49 for details of our exciting free to enter Micronet 800 competition.

Electronics & Computing Monthly is normally published on the 13th day of each month

EDITORIAL

Despite the fact that the price/performance, and indeed size/performace, ratios of computers have altered drastically over the past decade, the basic configuration of a computer has remained much the same. Information to a system is input, at least initially, via a keyboard and the output provided via a VDU screen or, if a permanent record is required, in the form of the printed word. If a computer programmer of the sixties has spent ten years stranded on a desert island, they would have little difficulty in recognising a BBC micro and assorted peripherals for what it is.

Over the next decade however, it's likely that major elements of a typical computers system will undergo radical change.

Key To Success

In striving for ever smaller computers one of the major sticking points is the physical limitation imposed by the keyboard. The obvious way to overcome this restriction is to replace the keyboard by some form of voice recognition facility. Work in this area is taking place in every country with an interest in computer R & D and, although at present, a reliable system capable of coping with the wide variety of human speech variations, is some way off, increasingly sophisticated systems are continually being demonstrated.

Output Options

When it comes to outputting data from a system, computer speech generation is already at a fairly advanced stage over and refinements over the next few years will provide any computer with a speech capability indistinguishable from a human being.

Speech, though, is a very transitory form of communication and in many cases some form of visual output will be required. Our news pages this month give details of a new flat screen LCD television that shows the way for VDU's of the future.

The current in work – networking – is also an era in which the computer of the future will feature considerable capability. The increasing use of satellites will mean that, as advances in RF technology procede, most computers will be part of some sophisticated radio network. This will endow even the smallest machine with vast processing and storage potential.

Within a wrist watch sized computer, with high definition colour display that is capable of reacting to spoken commands and responding with either a spoken message or detailed visual output. The machine will be in continual contact with a network and allow its user to gain instant access to almost any item of information desired.

What price our man from the sixties recognising this machine?

Down To Earth

On more immediate matters, any of you who have developed any software of a technical/scientific nature should turn to our letters page for news of a new feature we plan to introduce to the pages of *E& CM* over the next few months.

We're asking for those of you who have produced a package of a specialised nature to write and let us know of your work. No matter whether you think it to be too specialised to be of interest to other readers, you would be surprised at the range of interests shared by them.

We look forward to hearing from you and hope that our new feature will stimulate interest in the area of specialised software.

May 1982
Electronics & Computing Monthly
Britain's First Electronics & Computer Applications Magazine
INTERFACING THE BBC MICRO
UNDERSTANDING DATA STRUCTURES
HIGH RESOLUTION GRAPHICS
AIDED DESIGN
KIDS CORNER
24-TUNE DOORBELL
DATA TRANSFER BETWEEN MICROS
TI-99/4A Home Computer PULL-OUT Section

MAKE SURE OF GETTING YOUR COPY!
HAVE IT DELIVERED EVERY MONTH

"Electronics & Computing Monthly" has become required reading for thousands of Engineers, Technicians, Managers, Company Directors, Teachers, University Lecturers and Students plus many other electronics and computing enthusiasts. The magazine has a unique blend of theory, projects, software and technical features supported by technology updates — science and general features on a wide variety of high technology topics. If science, electronics and computing are your areas of interest then you can guarantee a good read in "Electronics & Computing Monthly". Special product features each month will combine interesting articles on a particular product theme with special promotions by advertisers. Special products on offer will beat all others on price. Scopes, digital multimeters and home computers are among the variety of goods which will be available at new low prices throughout the year.

SEND TO

ELECTRONICS & COMPUTING MONTHLY

**SUBSCRIPTIONS DEPT
COMPETITION HOUSE
FARMDON ROAD
MARKET HARBOROUGH
LEICESTERSHIRE**

Please send Electronics & Computing Monthly for the next 12 issues to commence from . . . . .Issue.

I have enclosed a cheque/Postal order for **£9.50** (UK only)
£13.95 (Overseas)

Name . . . .

Address . . . .

Town . . . .

Atari – New Prices

Atari International (UK) Inc has announced price amendments for their home computer products.

Following the favourable response to the upgrading of the Atari 800 home computer from 16K to 48K RAM earlier this year, Atari International are able to offer the 800 home computer, including Basic Cartridge and manuals at £299.99.

Atari are also offering a limited special promotion for the Atari 400 home computer. The recommended retail price of £149.99 will include the full Programmer kit – Basic Cartridge, The Atari Basic Reference Manual and The Atari Basic Self-Teaching Guide providing everything you need to sit down and start programming your computer.

QUICKSILVA Add ZAP!

QUICKSILVA, now established as the No. 1 UK Software House with a range of 37 top-selling programs, has appointed ZAP! UK as their distributor to UK specialist computer shops.

According to Neil Johnson, Managing Director of ZAP!, the idea behind the new arrangement is simple:

"QUICKSILVA make the best software on the market. ZAP's job is to deliver it, fast!"

Combined with a 6 day a week telesales service, ZAP! offers 48 hour delivery Nationwide via Courier Express.

In addition, ZAP! supplies a monthly bulletin to all specialist computer shops, with news of new QS programs and promotions.

For further information, contact:
Neil Johnson; ZAP! UK
PO Box 14
Hounslow
TW3 3PL
Tel: 01-572-1911

Essex Micro Electronics

Essex Micro Electronics are a newly formed company that intend to provide hardware add-ons for Jpiter Ace Microcomputer.

For a copy of their present and future products list send an SAE to the address shwon below. The Company alos intend to provide, of course, a certain amount of utility software to use with the add-ons.
Essex Micro Electronics
4 Hatch Road
Pilgrims Hatch
Brentwood
Essex
CM15 9PX

Oric in Japan

Oric Products International have announced that a new company is being formed in Japan to sell the new range of home and personal micros and peripheral equipment. Oric Japan will manufacture and market those products exclusively in Japan and competitively in South East Asia.

The new company is a joint venture between Oric Products International and the team in Japan, which estimate sales of the basic Oric to be 120,000 in a 12 month period commencing July 1983.

Oric Japan already have sales outlets lined up to take the Oric in good quantities.

Oric are currently the third best selling computer in the UK and are selling high volumes of products throughout Europe.

New Generation Software

The latest addition to the range of arcade games written by Malcolm Evans is now available through branches of Boots or direct from New Generation Software. The latest release in question is Knot in 3D, which has been written for the 48K Spectrum and which can be played with Kempston, AGF or Mikrogen Joysticks without modifications.

This is a full action graphics game that needs nerve and quick reactions! Hurtling through a void your task is to travel as long as possible, scoring points along the way. Be careful though as you will have to avoid your own trail and those of up to four chasers. As trails are left you will have to thread your way through, but don't forget – the more trails the less space for manoeuvring. You will be terminated when you finally get through the 'The Knot!'. The game retails at £5.95.

Spectrum Rock

The latest album from Pete Shelley – the driving force behind the Buzzcocks punk group – may not at first sight appear to be of any interest to readers of *E&CM.* The record is however one of those following the latest trend in the record industry, and incorporates a program for the Spectrum computer.

The program is loaded into a 48K Spectrum and the routine run in sync with the music to provide a visual display to augment the music.

Next month we'll be interviewing Pete Shelley and looking at this new trend in music publishing in a little more detail.

New ZX81/Spectrum Software

In a further expansion of its ZX81 and ZX Spectrum software library, Sinclair Research has launched nine new cassette programs, including an additional language, FORTH, and advanced chess game and another adventure for computer 'cult-figure', Horace.

For ZX81 and Spectrum users with programming interests the new FORTH program offers a languages combining BASIC's simplicity with the speed of machine code. FORTH is extendable by user-defined commands, and its compiled code occupies less than a quarter of the equivalent BASIC program, yet runs up to ten times as fast.

A major challenge for the games enthusiast is the new eight-level Cyrus-IS-Chess (for Spectrum only, by Intelligent Software). Based on the 'Cyrus' program, which won the second European Microcomputer Chess Championship, and was also the first to defeat the Cray-1's chess program at speed chess, its features include cursor movement to pieces, replay and 'take-back' facilities and the ability to function two-player game board.

Three further Spectrum games' programs are also available: a new Horace adventure with The Spiders, an over 10,000-word Scrabble and a full feature Backgammon. For the ZX81, Sinclair has introduced two new adventure games, Sabotage and City Patrol.

Finally the growing demand for business software is met by the new Small Business Accounts program (48K Spectrum only), which provides balance sheet and profit and loss information together with VAT returns.

Ranging in price from £4.95 to £15.95 inc VAT, all the new cassettes will be available in the next two weeks mail order from:
Sinclair Research
Stanhope Road
Camberley
Surrey
Tel: 0276 685311
and shortly from major branches of W H Smiths, boots, Currys, John Lewis Partnership, House of Fraser, John Menzies and Greens, and other leading chains and computer stores.

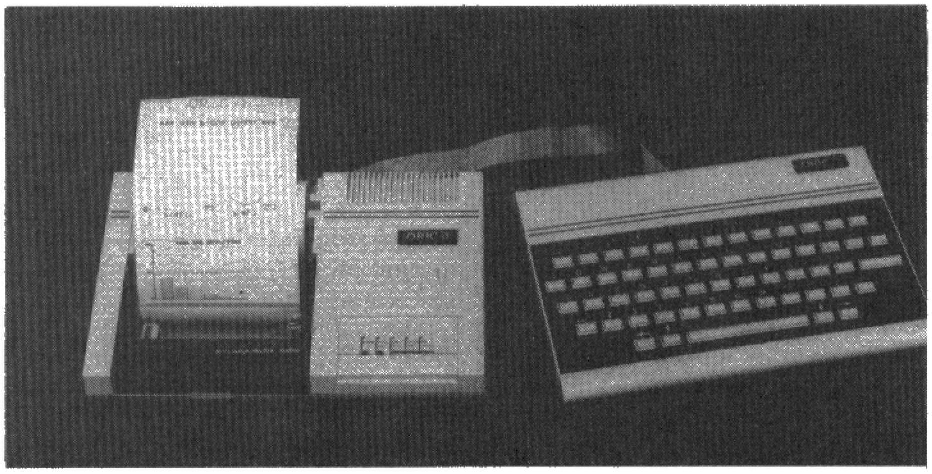
Pocket Colour TV

The first-ever LCD pocket colour television in the world has been eveloped in Japan by the Epson Corporation and Suwa Seikosha Ltd, the parent company of Epson (UK).

Measuring 16cm x 8cm x 2.8cm, the pocket TV utilises new picture display devices developed by Seiko in its development of a TV watch.

Amongst the pocket TV's advantages are no colour aberration at corners, or distortion of pictures. The display is of the illuminated type for maximum visibility in dark or light situations.

Commenting on the development, Mr. J. Patterson, of Epson (UK) Ltd. said: "Whilst there are no plans to have the new television available in the UK, it reflects the continuous research and development being undertaken in Japan.



Take a Memo

A British-designed electronic memo pad which can carry out the functions of a calendar, diary, address book, note pad and expense account log has been launched into the UK by Domicrest Ltd. The unit measures 136mm (W) x 90mm (H) x 9mm (D), about the same size as a cigarette case. Called the Biztek Pad it will be in the shops in late June with a retail price of £69.95.

Comprising a potential memory of 4000 characters the Biztek Pad is battery operated and combines a 43 key push button keyboard with a large LCD with 16 alpha-numeric characters, 20 numeric and clock digits and 45 special symbols.

Oric Plot

Oric Products' first peripheral for the Oric 1 – the specially designed four-colour plain paper printer – will be available in retail outlets from 1 June onwards at £169.95 (inc VAT).

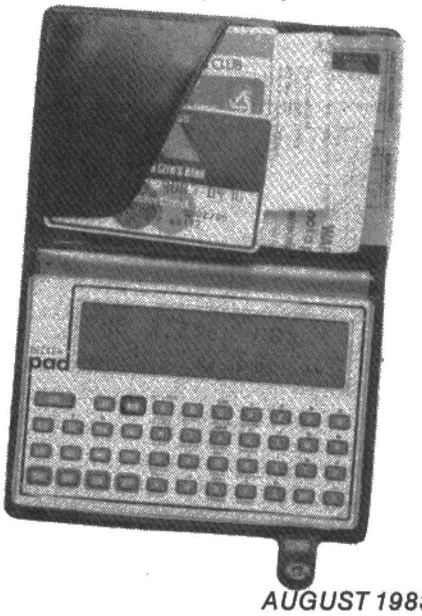
The printer, which features the Alps mechanism and an internal power supply, comes complete with a connecting lead. No other accessories are required. The Oric Colour Printer k(MCP40) is plugged directly into the Oric expansion port and can be used with both 48K and 16K models.

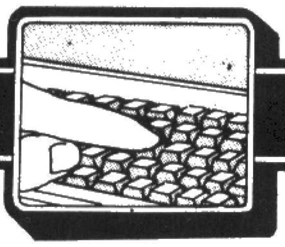
The four colours – black, blue, red and green – are provided by rolling ball point pen dispensers. The printer features full alphanumeric capability and graphics specification. Sample programs are included in the comprehensive manual.

ROM Box

ROM-81 is a memory expansion unit for the ZX81 personal computer which enables the user to read useful routines and commonly used information, stored in UV erasable, programmable Read Only Memory. The unit is supplied with EPROMs as these are normally programmed and provided by the user.

Two pin sockets allow either 2726 or 2732 EPROMs to be used. They can provide up to 8K byte and 16K in the ZX81 memory map, which is just below the Basic area. Separate 2K and 4K decoding is link selectable to make it possible to vacate locations occupied by other peripheral cards.





Under Pressure

NTC Services of Liverpool have introduced a series of low-priced transducers for interfacing directly to the BBC model 'B' Microcomputer. These devices are self-contained and are provided with the necessary connections for the on-board power supply and 'ADVAL' ports. The range currently consists of the following three models and others will be introduced in the near future. 1 Pressure transducer model 'P1' is a Bourdan tube based device available to cover PSI ranges 0-30, 0-60, 0-100, and 0-200. The gas connection is a tapered ¼" BSP male and compression couplings to suit any type or size of tube can be supplied on request.

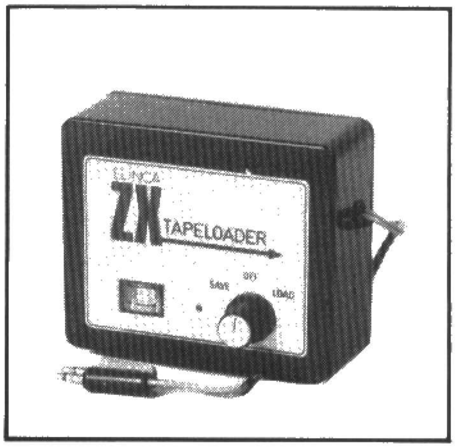
2 Linear Displacement transducer model 'M1' capable of resolving displacements of less than 1mm. This has been designed specifically as a teaching aid and experimental tool to demonstrate and investigate such physical phenomena as Young's Modulus, Thermal Expansion, Hooke's Law and Levers etc.

3 Model 'M2' which is similar to Model 'M1' but with the displacement-head sited on a separate board. This will be found useful where space is restricted.

Software on Cassette and Mini Floppy-disc is available for use with these devices providing various impressive displays for pressure, velocity, weight and displacement experiments etc. The manufacturers can undertake to write special software to customer requirements and modify the hardware for use with other types of computers.

These transducers are almost frictionless and will find wide application not only in Education but also in Research, Industry and Process control.

Full details from:
NTC Services
331 East Prescot Road
Liverpool
L14 2DD
051 228 4690

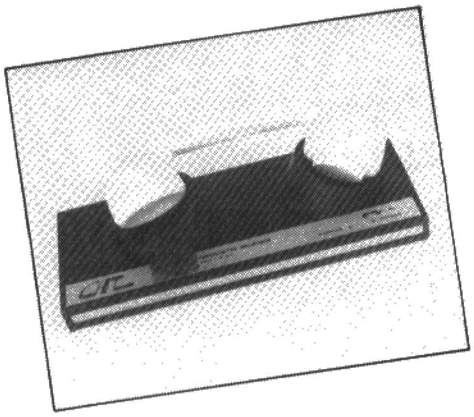


Tapeloader ZX81

Elinca Products Ltd have announced the introduction of a tape loader for use with ZX81 computers. The ZX tapeloader has been developed to solve problems of computer loading and saving, particularly when using a standard tape recorder.

The ZX tapeloader filters and stabilises signals in both directions, providing a constant signal that is matched to the computer. A major problem encountered when using a standard tape recorder for loading, due to lack of volume measurement and control, when the recorder is switched to 'play', is eliminated with the ZX tapeloader, which incorporates an audio output indicator and signal amplifier to enable the correct computer to be used.

The ZX tapeloader retails at £14.99 and details can be obtained from dealers or direct from the manufacturers:
Elinca Products Limited
Lyon Works
Capel Street
Sheffield
S6 2NL
Tel: 0742 339774



Modem from DaCom

DaCom, Milton Keynes, has brought out a new high quality acoustic coupled modem for international communications. The new coupler, the CCITT CAT V21, allows users to access remote computers by telephone without any wired connection to the instrument.

The CCITT CAT can communicate with any V21 compatible modem. It operates a 300 bps to the international V21 standard in both originate call and answer mode. A self test facility is built in for rapid fault isolation.

This new modem is to cost £198.
Further information from:
DaCom Systems
200 Conni Burrow Boulevard
Milton Keynes
MK14 7AH

Speak Up

Timedata are introducing another add-on for Sinclair computers; the ZXS Speech Synthesiser.

Retailing at £32.50 inclusive of VAT and p&p, it is one of the cheapest synthesisers available.

Based on the SP-0256 chip, which produces 'allophones' – the basic components of speech – the ZXS can be programmed to produce virtually any English word.

The ZXS can be used with either the ZX81 or the Spectrum.

Timedata Ltd
16 Hemmells
High Road
Laindon Basildon
Essex SS15 6ED

Stacks of Cartridges

A range of VIC-20 cartridges were recently introduced by Stack. The cartridges available are Super Graphics from VICKIT 1, the Best of Both on VICKIT 2, Hi-speed cassette operation PLUS machine code monitor on VICKIT 4, VICKIT 5 the 6502 Assembler, Supercharger Plus.

For the economy minded user, the cartridges are available in EPROM versions and can be fitted inside the MULTI ROM CARRIER which is an empty cartridge with sockets to take up to 4 EPROMS.

All the cartridges can be fitted into the 4 SLOT MOTHERBOARD (shown in the centre of the display). A IEEE-488 cartridge for the VIC-20 has also been released by the company. This allows VIC-20 owners to use Commodore disc drives and printers with their VIC.

The benefits for users such as schools are tremendous, particularly as it means that many PET programs will run on VIC with only minor changes.

The cartridge costs £39.00 plus VAT.
Further details are available from:
Stack
290/298 Derby Road
Liverpool 20

ZX SOUND BOARDS

Jim Cowie, D. O'Donnell and J. McAinsh describe the operation of the AY-3-8910 Programmable Sound Generator and describe their ZX81 and Spectrum sound boards.

There is an increasing trend for computers aimed at the home market to feature a sophisticated sound generation capability as part of their specification. The Sord M5 (reviewed last month) is just such a machine and depends for its excellent sound effects on a dedicated Programmable Sound Generator. In the Sord's case, this happens to be a device from Texas Instruments yet the General Instruments AY-3-8910 has been around longer and arguably offers a superior performance.

Neither the ZX81 nor the Spectrum feature much in the way of sound capability, although the Spectrum makes a gesture in this direction with its BEEP command. The two projects presented here will rectify this situation and bring these computers up to date in terms of effects generation.

Before going on to describe the two projects in detail, we'll take a more detailed look at the AY-3-8910/AY-3-8912 itself. (Note the difference between the two devices is that the 8910 provides two output ports while the 8912 has only one I/O port).

PSG Overview

The AY-3-8912 contains 15 internal registers which control various aspects of the sound output, e.g. tone, attack, noise, etc. So very complex sounds can be synthesised quite easily without the need for extensive software. Also, once the internal PSG registers have been loaded, the controlling MPU (in this case a Z80A) is free to run away and tend to any other chores, until the registers once again need reprogramming in order to change the sound being generated.

Figure 1 shows the block diagram of the 8912. There are three independent tone generators and a noise generator. Any of the three tones can be enabled to its output. Noise can be added to any or all of the tones or it can be output instead of the tone. The amplitudes of the noise and tones can be set to any one of sixteen values or they can be varied by the envelope generator. The envelope generator amplitude modulates the outputs and can be set for fast or slow attack or decay, repeat or single shot.

The AY-3-8912 is a fifteen register device intended for use with CPUs which use a shared address and data bus and cannot be used directly with Z80 based computer but can be interfaced quite easily. This difficulty is overcome by using the data bus to carry data which the 8912 can use either as true data or as a register select depending on the state of the control lines. The function and states of the control lines are shown in **Fig 2.**

FUNCTION	BDIR	BC1
INACTIVE	0	0
CPU READ FROM PSG	0	1
CPU WRITE TO PSG	1	0
LATCH ADDRESS REGISTER OF PSG	1	1

Figure 2. The function and states of the PSG's register select lines.

All that is required, therefore, is to set the control lines to the correct configuration before sending the data on the shared bus.

The PSG Registers

In order to produce more subtle sounds it is necessary to be familiar with the functions of the individual registers. All the registers are 8 bit but some are cascaded to give better control over the function. The set of registers is shown in **Fig 3.**

Registers 0 and 1

These are cascaded to give a 12 bit word which sets the TONE PERIOD for channel A – only the 4 least significant bits of register 1 are used. The registers can be set to any value between 1 & 4095 decimal. Since the clock is pre-divided by 16 before being fed to the tone generator the output frequency is:-

$$\text{output Frequency} = \frac{\text{F clock}}{16 \times N}$$

-: where N lies between 1 & 4095

Registers 2 & 3, 4 & 5

These are similar to 0 & 1 and control tone generators B & C respectively.

Register 6

This register is used to control the pseudo-random noise generator. Only the 5 least significant bits are used. This register too has its clock frequency pre-divided by sixteen.

Register 7

This is perhaps the most important register – the master enable. It is organised to be active low so that placing a 0 in the register will enable all channels and 255 will turn them all off. 254 will enable Tone A only. If bit 3 is also set low with 246 the noise generator will be mixed in with Tone A. Similarly bits 1 & 4 control Tone B & noise Band bits 2 & 5 control Tone C & noise C.

Register 8

This register is used to set the amplitude of Channel A in the fixed level output mode. Only the four least significant bits are used allowing the output to be set to one of sixteen fixed levels. If bit 4 is set to a "1" the output level is set by the envelope generator and thus bits 0 to 3 have no effect.

Registers 9 & 10

These are similar to Register 8 and enable Channels B & C respectively.

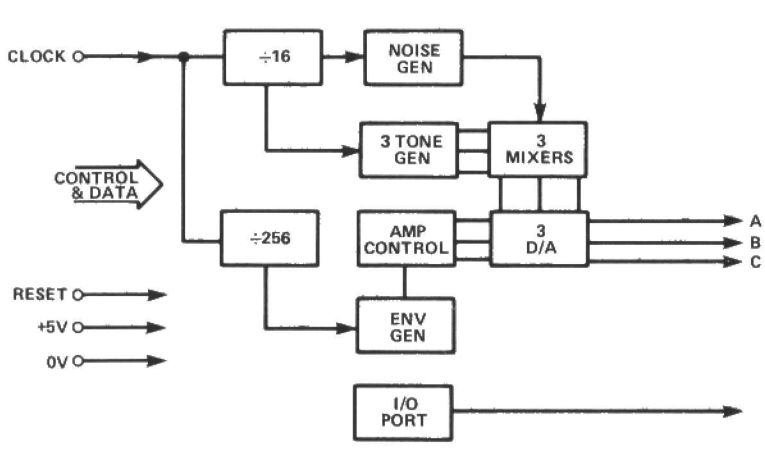


Figure 1. Block diagram of the 8912 PSG. the IC features three independent tone generators and a noise generator.

REGISTER	FUNCTION	BIT							
		7	6	5	4	3	2	1	0
R0	CHANNEL A TONE PERIOD	8 BIT FINE TUNE A							
R1						4 BIT COARSE TUNE A			
R2	CHANNEL B TONE PERIOD	8 BIT FINE TUNE B							
R3						4 BIT COARSE TUNE B			
R4	CHANNEL C TONE PERIOD	8 BIT FINE TUNE C							
R5						4 BIT COARSE TUNE C			
R6	NOISE PERIOD				5 BIT PERIOD CONTROL				
R7	ENABLE	IN/OUT		NOISE			TONE		
			A	C	B	A	C	B	A
R8	CHANNEL A AMPLITUDE				ENV	4 BIT AMPLITUDE			
R9	CHANNEL B AMPLITUDE				ENV	4 BIT AMPLITUDE			
R10	CHANNEL C AMPLITUDE				ENV	4 BIT AMPLITUDE			
R11	ENVELOPE PERIOD	8 BIT FINE TUNE							
R12		8 BIT COARSE TUNE							
R13	ENVELOPE SHAPE					4 BIT CONTROL			
R14	I/O PORT	8 BIT PARALLEL PORT							

Figure 3. Details of the AY-3-8912's 15 eight bit registers.

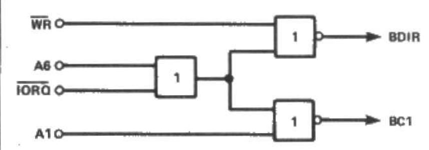


Figure 4. Detail of the Spectrum Board's decoding circuitry. This uses the IORQ signal together with the required port address.

189 sets both BDIR & BC1 to "1" to latch the address register. OUT 191,B will then place the data – B in the selected register by making BDIR "1" & BC1 "0". IN 189 can be used to read the contents of the register last pointed to by an OUT instruction. This could be used to record register contents if a particular sound had been generated randomly or the contents of register 14 could be read thus receiving data from the outside such as a joystick. The use of these commands will be illustrated later in example programs.

Registers 11 & 12
These two registers together form a 16 bit word which sets the envelope period. The clock is pre-divided by 256 so a 2MHz clock will give a period range from about 8kHz down to 0.1Hz.

Register 13
This register gives the cycle/shape of the output. The envelope generator further divides the envelope period by 16 to produce a 16 state per cycle envelope pattern as selected by bits 0 to 3. The desired shape is attained by defining count up/count down, single shot or repeat. The patterns available are set out in **Fig 9** along with their decimal equivalents to make its use easier.

Registers 14 & 15
These registers are in/out ports whose function is controlled by bit 6 of register 7. "0" for input, "1" for output. (Register 15 only on AY-3-8910).

Spectrum Sound Board

The circuit diagram of the Spectrum sound board shows that, apart from the AY-3-8912 the IC's clock and the audio output stage, the only other components concern themselves with the decoding operation. The decoding is greatly simplified on the Spectrum by using the IORQ signal together with the required PORT address. Thus OUT 189,A will prepare the PSG for data transfer to register A. The decoding of the WR, IORQ, A6 & A1 to BDIR & BC1 is shown in **Fig 3**. OUT

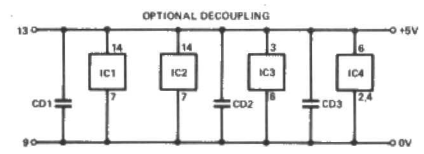


Figure 5. The optional supply decoupling capacitors.

Construction

Following the PCB and overlay shown, construction should pose no problems. When the unit has been built and checked for shorts it can be fitted to the SPECTRUM edge connector, and power applied. The PSG will reset itself and the SPECTRUM should power up in the normal way. If not, remove board and check for any mistakes.

SEE E&CM MAY 1984 P.13 OF
FOR ERROR/ CORRECTIONS

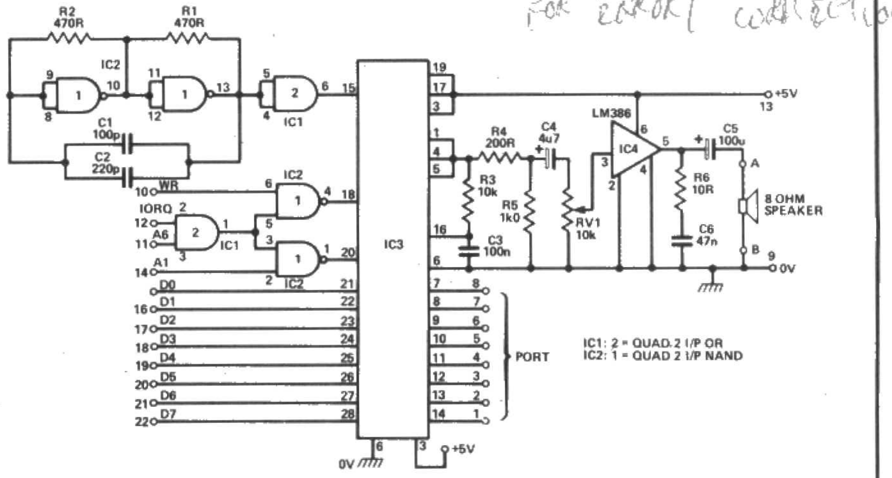


Figure 6. Full circuit diagram of the Spectrum Sound Board. Apart from the oscillator and decoding circuitry, the board features an audio power amplifier.

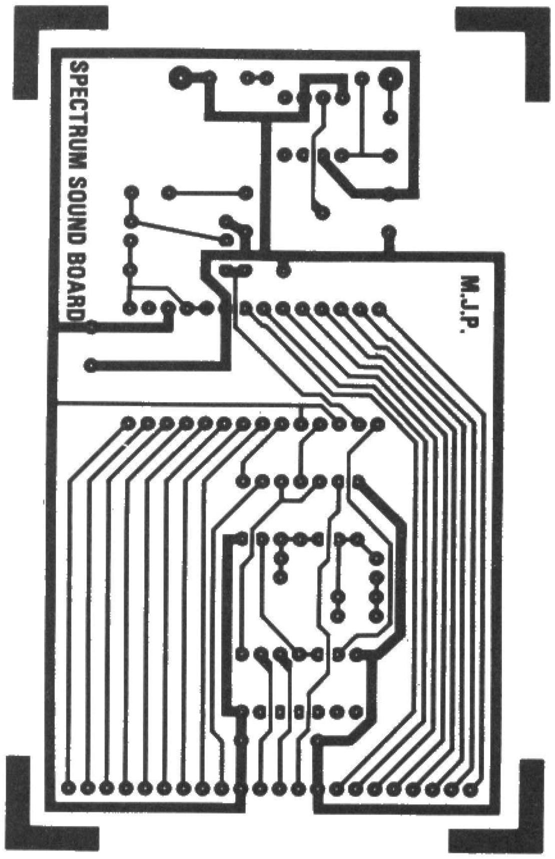


Figure 7. The foil pattern of the Spectrum Board shown full size.

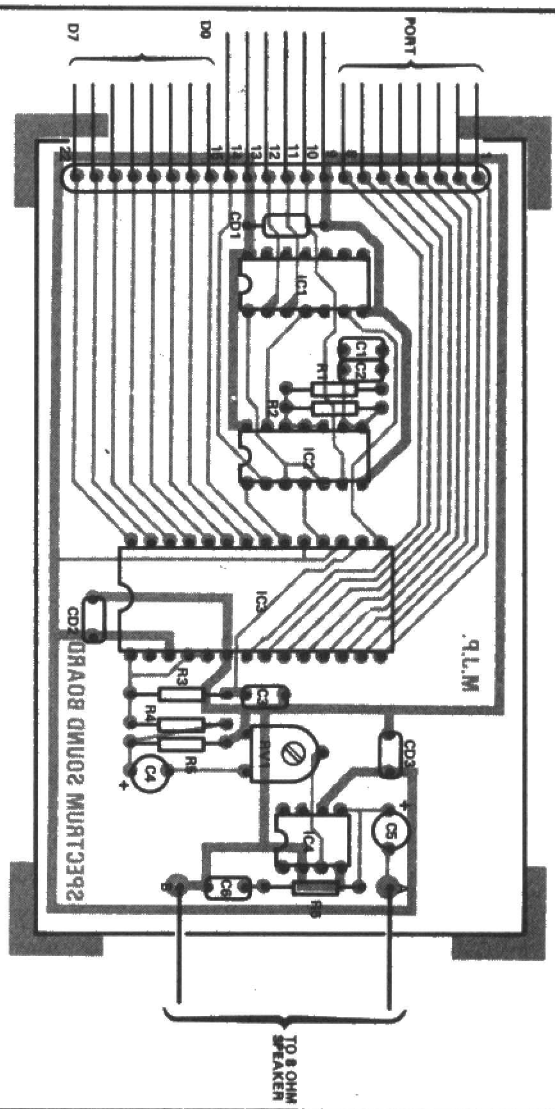


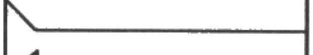
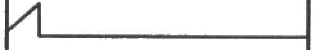
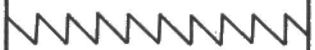
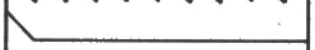
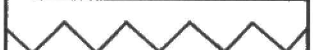
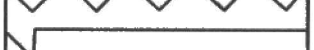
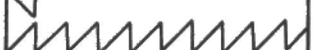
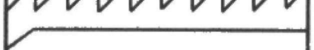
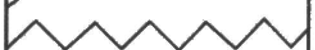
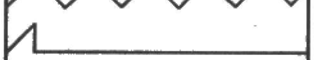
Figure 8. The overlay of the Spectrum Sound Board.

PARTS LIST

Resistors	
R1,2	470R
R3	10k
R4	200R
R5	1K
R6	10R
RV1	10k
Capacitors	
C1	100p
C2	220p
C3	100n
C4	4u7
C5	100u
C6	47n
Semiconductors	
IC1	7402
IC2	7432
IC3	AY-3-8912
IC4	LM386

Miscellaneous
PCB, connector, speaker, DIL sockets, etc.

R13 BITS

	B3	B2	B1	B0	
DECIMAL	CONTINUE	ATTACK	ALTER-NATE	HOLD	GRAPHIC REPRESENTATION OF ENVELOPE GENERATOR
0	0	0	X	X	
4	0	1	X	X	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

DURATION OF ONE CYCLE

Figure 9. The bit patterns associated with register 13, the envelope controller.

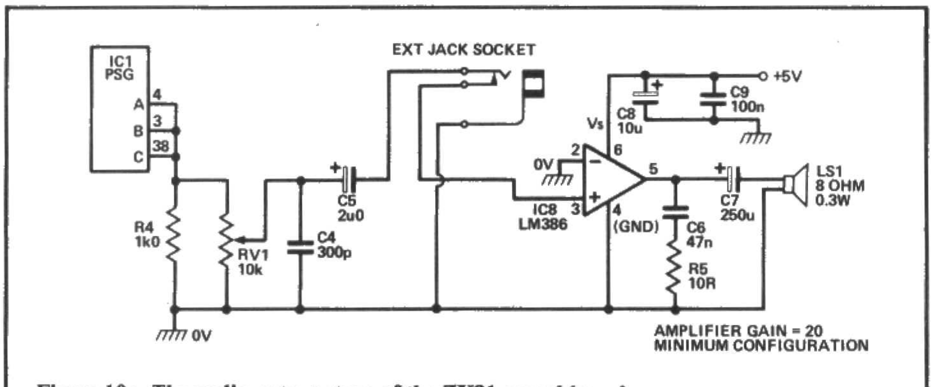


Figure 10a. The audio output stage of the ZX81 sound board.

ZX81 Sound Board

Even though the Z80A used in the Sinclair ZX81 has a facility for I/O commands, it was decided to place the sound board within the computer's available memory map. This allows the use of BASIC memory commands like PEEK and POKE to monitor and load the sixteen registers within the PSG. In order not to interfere with RAM, the soundboard is memory mapped within the "image copy of ROM" area which is disabled whenever addressing the board. The addresses used are:

LATCH ADDRESS-----8192 (decimal)
READ/WRITE ADDRESS-----8448 (decimal)

Circuit Description

Figure 10a, b shows the circuit diagram of the sound board. The PSG is permanently enabled by tying A8 (pin 25) high and A9 (pin 24) low. Access to the PSG bus is by activating the BUS CONTROL LINES BC1 and BDIR; this is acceptable because DA0 to DA7 are controlled high impedance I/O lines. Capacitor C1 provides a power-up reset to the PSG which initialises all the internal registers to "0".

The PSG requires a clock frequency between one and two MHz, but as the ZX81 uses a clock of 3.25MHz it is necessary to divide it by two. This is done by passing the clock through IC6a giving an output of 1.625MHz which is fed to the PSG.

Because the Z80A is running too fast to allow it to read and write from and to the PSG, two WAIT states have to be introduced into the timing cycle at the appropriate points. The WAIT states are generated by gating a high input through inverter IC5f onto the WAIT state line (19A on the ZX81 connector). When A13 goes high it disables the ROM C/S line on the ZX81 via D1, so preventing any conflict between the sound board and the ROM operating system.

IC2,3,4 act as address decoders, both to activate BC1 and BDIR at the correct time and also to control when the WAIT states are generated. C2,R1,R2 and IC4c form what is known as a half monostable, so that when pin 6 of IC2b (line marked W) falls low then IC4c outputs a short low pulse clearing IC7 which is part of the WAIT state generator.

IC8 is a low power audio amplifier, the LM386. The analogue outputs A, B and C from the PSG are tied together and are fed to the input of IC8, the output of which drives an 8 ohm, 0.3 watt speaker. VR1 acts as a volume control, and the jack socket allows you to connect an external amplifier while switching off the input to IC8.

E&CM

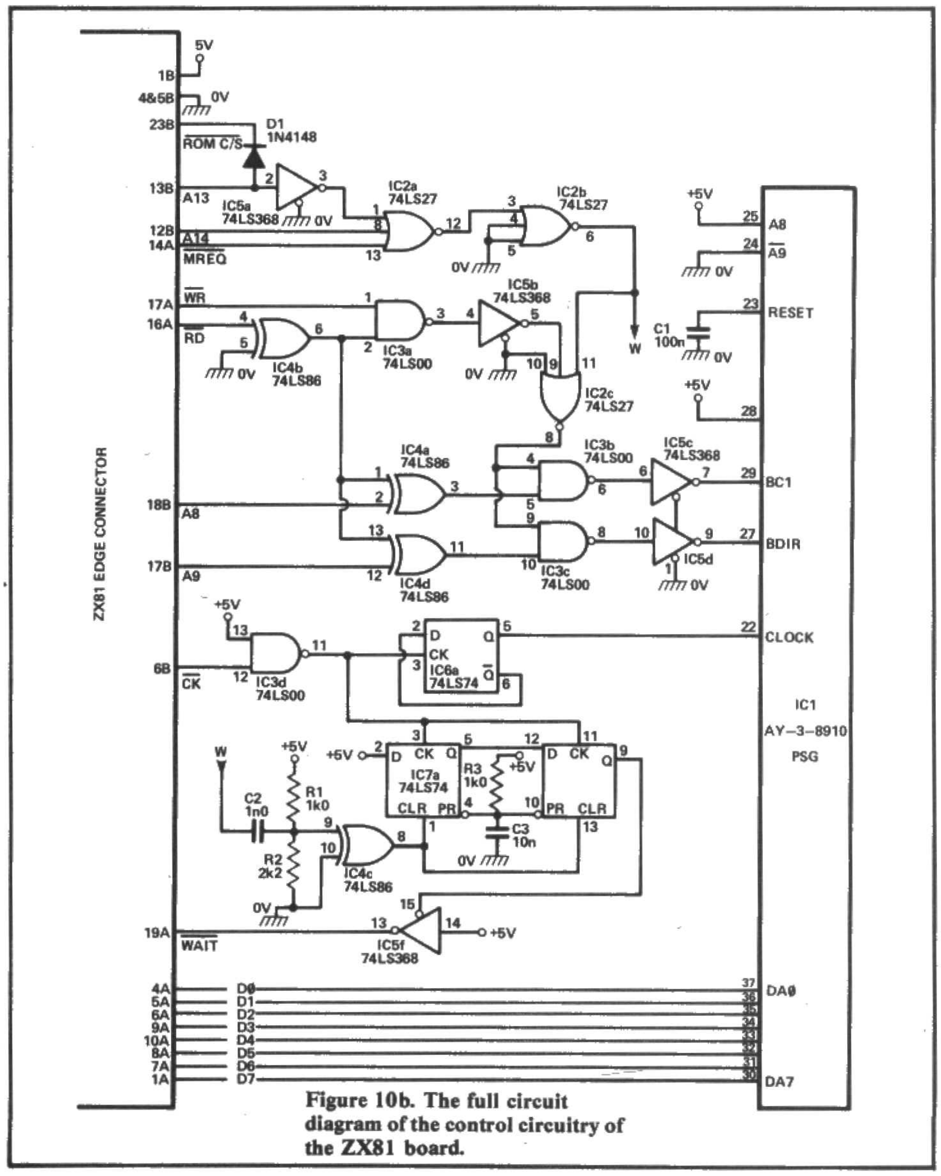


Figure 10b. The full circuit diagram of the control circuitry of the ZX81 board.

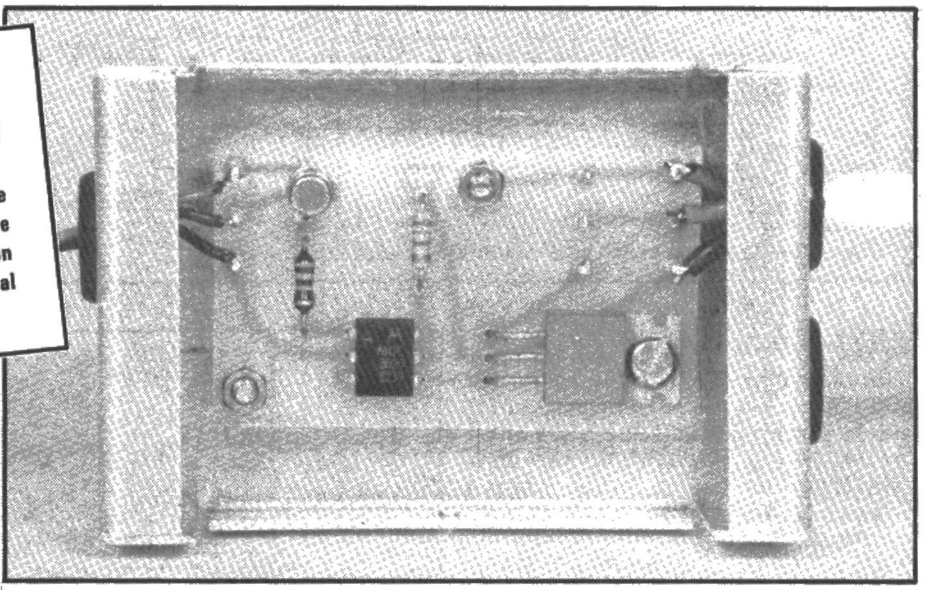
Next month: constructional details for the ZX81 board plus software for both designs.

NEXT MONTH

MONITORS

To tie in with our feature on video techniques on the ZX computers, we look at the wide range of monitors suitable for use with computers.

We explain the difference between composite drive waveforms and RGB inputs, examine the various tube technologies and their effect on ultimate resolution and delve into the internal workings of a typical monitor.



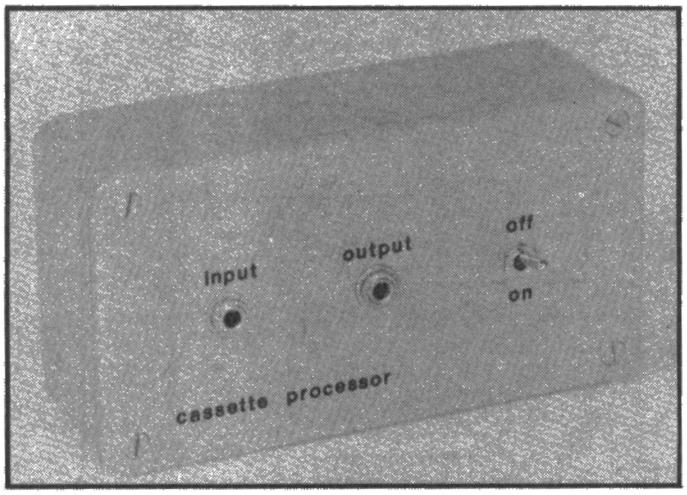
ZX VIDEO INTERFACES

Both the ZX81 and the Spectrum provide a UHF modulated video signal but fail to provide any composite video output or, in the case of the Spectrum, an RGB output.

Next month we look at the topic of video signals in general and provide some pointers to obtaining higher quality displays from the two ZX machines.

COMPUTER CONTROLLED ENLARGER TIMER

Designed for the BBC computer, this project describes both the hardware of the isolating interface (a trial based design) and the software that turns the computer into a sophisticated darkroom print timer.



ATARI 400 & 800

Recent price cuts have once again focused attention on these two popular computers from Atari.

For those of you who are unfamiliar with the features of these machines our review will reveal that they are, possibly, the dark horses of the home computer world.

SCOPE REVIEW

We test a popular dual beam 'scope. For any serious design work a 'scope is an essential piece of equipment and the model we test offers excellent performance at a modest price.

PLUS – Data on an attractive A/D device, a look at the Dragon's sound capabilities, an expanded software review section and all our regular features.

ORIC SIGNAL PROCESSOR

Some people have reported problems when loading and saving Oric programs on their cassette recorders – particularly so when using the 2400 baud speed.

Next month we publish a circuit that should solve any problems by 'cleaning up' the cassette players signal before it enters the Oric.

On Sale AUGUST 13th

Articles described here are in an advanced state of preparation however circumstances may dictate changes to the final contents of the issue.

THE BBC ASSEMBLER

Mike James shows that the BBC micro's assembler, although quite simple, really is a very clever tool.

One of the great attractions of the BBC Micro to anyone technical, is its in-built 6502 assembler. Using this, 6502 assembly language can be written as conveniently as BASIC routines. The machine allows BASIC and assembler instructions to be mixed, and writing most of a program in BASIC, just using assembly language subroutines to speed things up, must be the most efficient way of producing fast programs, quickly. Even with these facilities though, there is a tendency for advanced programmers to belittle the 'tiny assembler' offered by the BBC Micro and compare it unfavourably with other 'assemblers-they-have-known'. The reason for some of the less than flattering comparisons lies in the fact that an assembler is a more than straightforward piece of software capable of translating commands like LDA etc. to the correct 6502 op codes. A program that does just this would be called an assembler but in general even the simplest assembler is capable of keeping track of the addresses that each label used in the program represents. For example, if the start of the program was at &100 (in this article the standard BBC symbol & for hexademimal numbers will be used) then the simplest assembler would cope with JMP START as long as the value of START had been defined to be 100 somewhere in the program. An assembler that will handle labels in this way is usually considered to be necessary for any sort of assembly language programming involving more than a very few instructions. In the professional programming world however, assemblers are generally even more sophisticated than this. There are two extra facilities usually considered to be important if any 'serious' assembly language programming is to be undertaken.

1 — conditional assembly
2 — macros

A conditional assembler allows a number of different versions of a program to be written as one text file. For example, suppose a program has to have slight changes made to it depending on the exact machine configuration it will run on. Without a conditional assembler complete copies of each version would have to be kept although each may be only slightly different. Imagine the problem of trying to make a small change to the program – it would have to be made to each version separately! Using a conditional assembler one copy of the program can be saved, complete with all the different changes necessary to produce the different versions. When the program is assembled it is only necessary to specify which version is required and the assembler produces it. There is only one program to update and look after!

Whenever assembly language is written it is usual to find that the same combination of instructions crop up time and time again. One solution to this problem is to collect the groups of instructions together as subroutines. This is often not the most useful solution for assembly language programming however as these programs are often used for speed and it is therefore desirable to avoid as much jumping about as possible. What is needed is some way of grouping together instructions and, instead of jumping to them like a subroutine, having them included in the program whenever the collection is referred to. This way of collecting instructions together for insertion into a program is known as a 'macro'.

What is surprising is that the 'simple' BBC assembler can be used for conditional and macro assembly! The BBC assembler is, as suggested by critics, very simple but, because of the way it is integrated within the BASIC interpreter, it can share much of the power of BASIC. Indeed it is not too much of a claim to say that, contrary to the critic's view, this combination of simple assembler and advanced interpreter is a very clever way of doing things.

Using the Assembler

To leave BBC BASIC and use the assembler is simply a matter of enclosing any assembly language statements by square brackets. The rule is that anything within square brackets will be treated as assembly language, anything outside square brackets is pure BASIC. This is a remarkably easy way of switching between BASIC and assembler but it is important to understand the different ways that BASIC and assembler are processed. When the BASIC interpreter encounters a line of BASIC it carries out the instruction but when the assembler meets a line of assembly language it doesn't carry out the instruction, it simply translates it to machine code. For this machine code to be of any use it must be carried out sooner or later but this is nothing to do with the assembler!

When the assembler translates the assembly language it has to store it somewhere until it is needed. The problem of where to store the machine code is solved by two additions to BASIC. The first consists of a set of special variables, the resident integer variables, that are always present whether you use them or not and are always stored at fixed locations in memory. The resident integer variables are named @%, A% to Z% and the value stored in P% is used by assembler as the address at which the next item of machine code will be stored. In addition to this, every time the assembler stores an item of machine code it adds one to P% so making sure that each item of machine code is stored in a new memory location. By setting P% to a particular address before using the assembler it is possible to determine where the assembler stores the machine code that it produces. The second addition to BASIC is the provision of 'byte arrays'. BBC BASIC allows you to define special arrays that use one memory location per element, to define such an array a slightly modified version of the DIM statement is used:

```
DIM 'variable name' 'maximum index'
```

For example,

```
DIM CODE% 20
```

gives a byte array with 21 elements, i.e. element 0 to element 20. There is one other difference between a normal array and a byte array and this concerns the way that the 'variable name' is used. Instead of being the name of the array this is in fact a standard BASIC variable that is initialised to contain the address of the first element of the ▶

array. In other words, the byte array created by the last example isn't called CODE%, in fact it doesn't have a name in the usual sense, but the address of its first element is stored in CODE% and this can be used to gain access to it. It should be apparent that one way of making space for machine code is to define a byte array big enough to hold the code. To make sure that the assembler places the translated code into the array it is necessary to set P% to the start of the array. Consider the following short example.

```
10 DIM CODE% 20
20 P%=CODE%
30 [
40      LDA #65
50      JSR &FFE3
60 ]
```

The first two lines are BASIC. Line 10 creates a byte array with 21 elements and places the address of the first element in the variable CODE%. Line 20 stores this start address in P%. Lines 30 to 60 are assembly language. Line 30 is the opening square bracket that switches us from BASIC to assembler and line 60 is the closing square bracket that switches back to BASIC. The assembly language statement in line 40 loads the 6502's A register with 65 and line 50 jumps to a subroutine starting at &FFE3. To understand the program it's necessary to know that the subroutine at &FFE3 will print on the screen the character whose ASCII code is stored in the A register. Thus as ASCII code 65 corresponds to 'A' the program will print the letter A on the screen. However if the program is run it will only produce a listing of the machine code program looking something like –

```
1951 A9 41       LDA #65
1953 20 E3 FF    JSR &FFE3
```

remembering what the assembler does this won't be too surprising. Remember the assembler only translates the program to machine code and this is exactly what has happened in this example. Looking at the first line of the output, the first number, 1951, is the address where the first item of machine code, &A9, will be stored. The second item &41, will automatically be stored at 1952 and so on to 1955. To carry out the machine code the BASIC CALL command must be used.

```
CALL 'address'
```

transfers control from BASIC to a machine code routine starting at 'address'. Notice that CALL must be carried out by BASIC so the standard sequence in writing and using a machine code program is –

```
In BASIC – set up space for machine code etc.
In assembler – translate assembly language to machine code
In BASIC – use CALL to transfer control to the machine code
```

To use the machine code in the last example add line 70 –

```
70 CALL CODE%
```

(Notice the use of CODE% to give the address of the start of the machine code again!) Now when the program is RUN the program listing will be followed by a letter A and an error message. The reason for the error message is that the machine code routine doesn't contain a RTS (ReTurn from Subroutine) command to transfer control back to BASIC. Adding line 55,

```
55 RTS
```

solves this problem and emphasises how easy it is to modify a BBC machine code program!

Labels And Variables

So far the subject of labels has been ignored. In fact the BBC assembler has a very interesting way of handling labels. Although BASIC commands cannot be used within the assembler, BASIC expressions can! For example, instead of writing LDA #65 we could have saved ourselves the trouble of looking up the ASCII code for "A" by using LDA #ASC("A"). In the same way an expression can be used in the address field of an instruction. It is permissible to write JMP* 3+6, which would cause a jump to address 12 or even JMP CALL%+10 which would cause a jump to the address given by adding 10 to the contents of the variable CALL%. There are many other ways in which this powerful idea can be used and indeed one of the main problems is seeing which of the many ways is useful! The key point to remember is –

> You can use any valid BASIC expression, including defined variables, anywhere that the assembler would expect data or an address as part of the instruction.

The example 'print A' program can now be re-written in a more readable form as –

```
10 DIM CODE% 20
20 P%=CODE%
30 OUT%=&FFE3
40 [
50      LDA #ASC("A")
60      JSR OUT%
70      RTS
80 ]
90 CALL CODE%
```

Apart from the use of ASC("A") already described, notice the use of OUT% defined at line 30, to give the address that the JSR will transfer control to. In general, if you have any fixed subroutines or memory locations within an assembly language program it is better to define variables with the correct values and appropriate names within BASIC and then use these variables within the program. If this simple rule is followed, programs will be easier to understand and easier to modify.

The command JSR OUT% is using a normal BASIC variable as if it was a label. This is not unusual, after all a label is simply a variable that is used to hold an address. What is unusual is the fact that BBC assembler labels are also BASIC variables and this is one of the reasons that this simple assembler is so powerful! Using labels defined in BASIC is all very well but in most assembly language programs there is a need to refer to locations within the program. For example, if it's necessary to print a lot of letter As instead of just one the obvious thing to do is turn the previous example into a loop by jumping back to the JSR OUT% instruction. This address is likely to change as you develop the program however, and it would be a lot more convenient to write something like JMP LOOP% where LOOP% automatically contains the address of the JSR OUT% instruction no matter where it is. The assembler allows this to be done by adding one extra way of assigning a value to a variable. If an assembly language instruction is prefaced by a dot and then a variable's name, the current value of P% will be stored in the variable before the instruction is assembled. This means that the variable will contain the address that the instruction will be stored in. For example, try –

```
 10 DIM CODE% 20
 20 P%=CODE%
 30 OUT%=&FFE3
 40 [
 50          LDA #ASC("A")
 60 . LOOP% JSR OUT%
 70          JMP LOOP%
 80 ]
 90 CALL CODE%
```

The variable LOOP% is defined in line 60 to contain the address of the JSR OUT% instruction and this is used in line 70 to jump back to this instruction repeatedly. When the machine code is used by line 90 the result is that the whole screen is rapidly filled with letter As. It is important to notice that although LOOP% was set to a particular value by the assembler, it is still a BASIC variable and, once back inside BASIC, it can be used just like any other variable. If 85 PRINT LOOP% were added it would be possible to find out the address of the JSR instruction in line 60.

Two Pass Assembly

The only outstanding problem when using labels can be found by running the following program –

```
 10 DIM CODE% 20
 20 P%=CODE%
 30 [
 40    LDA #0
 50    BEQ EXIT%
 60    LDA -0
 70 . EXIT RTS
 80 ]
```

The program itself doesn't do anything useful, it loads the A register with zero then jumps to EXIT% if the result of the load was zero (it always is!). Line 60 is there just to fill the program out a little and line 70 is the instruction that returns control to BASIC. This program may not make very much sense but there is nothing wrong with it! However the BBC assembler gives an error message if you RUN it. The reason for this error message is not difficult to see. When the assembler reaches line 50 it needs to know the value of EXIT% but unfortunately EXIT% has not been defined and will not be defined until line 70. This is called the 'forward reference' problem and the traditional solution is to use a 'two pass assembler'. A two pass assembler, as its name suggests takes two 'looks' at the program. The first time through it picks up all the definitions of the labels and ignores any errors. The second time through it uses the values of the labels defined in the first pass to assemble the program correctly – any errors at this stage are real!

The situation with the BBC assembler seems hopeless – it's a one pass assembler. However this is where its close association with BBC BASIC comes to the rescue for the first but not the last time. There is an assembler instruction called OPT which can be used to suppress error messages and program listings. **Table 1** shows the various options.

n	OPT n
0	ignore errors and don't produce a listing
1	ignore errors but produce a listing
2	report errors but don't produce a listing
3	report errors and produce a listing

The default value for OPT is OPT 3 because all the programs that have been RUN so far have both reported errors and produced listings. If you look at the definition of OPT 0 the way to go about producing a two pass assembler should become clear. First run the assembler over the program using OPT 0 to define all the variables used as labels. As these variables are BASIC variables they still exist and their values are unaltered after the assembler has finished. At this point the assembler could be RUN again, this time using OPT 3 knowing that all the labels are defined. The only problem is how to run the assembler a second time? The entire assembly language program could be written out twice but this would soon put a stop to any serious applications! The answer is to use a BASIC FOR loop to pass the assembler over the program twice! This may sound uninteresting but it is our first example of using BASIC to modify the way that the assembler works. The two pass version of the forward reference example given above is:

```
 10 DIM CODE% 20
 20 FOR P=0 TO 3 STEP 3
 30 P%=CODE%
 40 [OPT P
 50    LDA #0
 60    BEQ EXIT%
 70    LDA #0
 80 . EXIT RTS
 90 ]
100 NEXT P
```

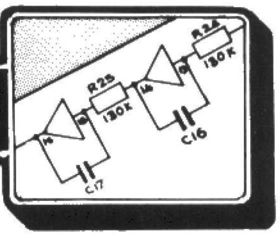
Even this example exhibits one subtlety. Care must be taken to set P% to the value stored in CODE% within the loop, otherwise the second pass over the program will store its machine code after the incorrect machine code produced by the first pass. Also notice the way that the assembler must be left before trying to execute the NEXT P instruction. Apart from very simple programs this two pass method of using the BBC assembler is the norm and for this reason it is all too easy to miss its implications for using BASIC with the assembler in other ways.

Conditional Assembly

We now have all the information necessary to see how to run the BBC assembler into a conditional assembler. For example, suppose a communications program that worked at two different speeds had been written. In any given application only the fast or the slow version of the program would be required and it would be possible to use something like –

```
IF FAST=1 THEN [LDA #600:] ELSE [LDA #300:]
```

where FAST is a normal BASIC variable that can be set elsewhere in the program. Depending on the value of FAST either the first instruction will be assembled into the program or the second. The only thing to be careful about is to remember to leave the assembler to execute the IF statement and then remember to return to it afterwards. ▶



As another example, it is often a good idea to use regular jumps to a subroutine that prints the values stored in certain variables while an assembly language program is being debugged. Once the program is working the obvious thing to do is to take out the debugging aids but this does cause something of a problem if another bug is then found. The best solution is conditional assembly.

```
10 TEST=1
.
150 [ LDA #&40
160   STA IRQC%
165 ]:IF TEST=1 THEN [ JSR DEBUG%:]
166 [
170   LDA AUXC%
180   ORA #&CO
```

This may look a little complicated but it is easy to follow. Line 165 first returns to BASIC to carry out the IF statement which will either re-enter the assembler to assemble JSR DEBUG% or just pass on to line 166 depending on the value in TEST. While debugging the program, TEST would be set to 1, otherwise to 0.

How to use IF . . THEN and IF . . THEN . . ELSE to conditionally assemble any list of instructions into a program should be apparent by now and so all that is left is to decide when the facility is useful and use it!

Macros

Now that we have seen how BASIC can be used to alter the way that the assembler works, the principles behind a macro are easy. Consider the fairly common problem of carrying out a shift or rotate instruction a few times. Normally this would be done by writing the command as many times as needed. For example, four arithmetic shift lefts would be –

```
100  ASL A
110  ASL A
120  ASL A
130  ASL A
```

Using the same idea as for the two pass assembler we could generate four ASL A instructions by placing a single ASL A instruction inside a FOR loop but this time not resetting the value of P% each time through. To take this idea one stage further we could write a PROC with an appropriate name and a parameter that would repeat the instruction a given number of times. That is –

```
1000 DEF PROCASL (N)
1010 LOCAL I
1020 FOR I=1 TO N
1030 [ ASL A: ]
1040 NEXT I
1050 ENDPROC
```

and in the main program the four ASL A instructions would be produced by

```
100 ]: PROCASL(4):[
```

which would first leave the assembler then call the PROC and then, after generating the required machine code re-enter the assembler. The PROC itself is fairly straightforward apart from the need to define I as LOCAL just in case it is used anywhere else.

Once this idea has been seen in operation it 'takes off' to produce all sorts of useful macros. For example, the parameters passed to the PROC both within BASIC can be used to control the assembler, i.e. as in the FOR loop or in conditional assembly or they can be used within the assembler as part of expressions. A general addition macro along the following lines could be produced:

```
1000 DEF PROCADD(N1,N2,ANS)
1010 [ CLC
1020   LDA N1
1030   ADC N2
1040   STA ANS
1050 ]
1060 ENDPROC
```

To add two numbers stored at memory location DATA1 and DATA2 and store the answer in DATA3

```
PROCADD(DATA1,DATA2,DATA3)
```

would be used. This would generate the correct machine code at whatever position within the main program it was used.

If a standard assembly language operation is available then the advantages of turning it into a macro are as great as turning it into a subroutine, however, as use of the macro generates the necessary machine code every time it is used there are none of the disadvantages of using a subroutine!

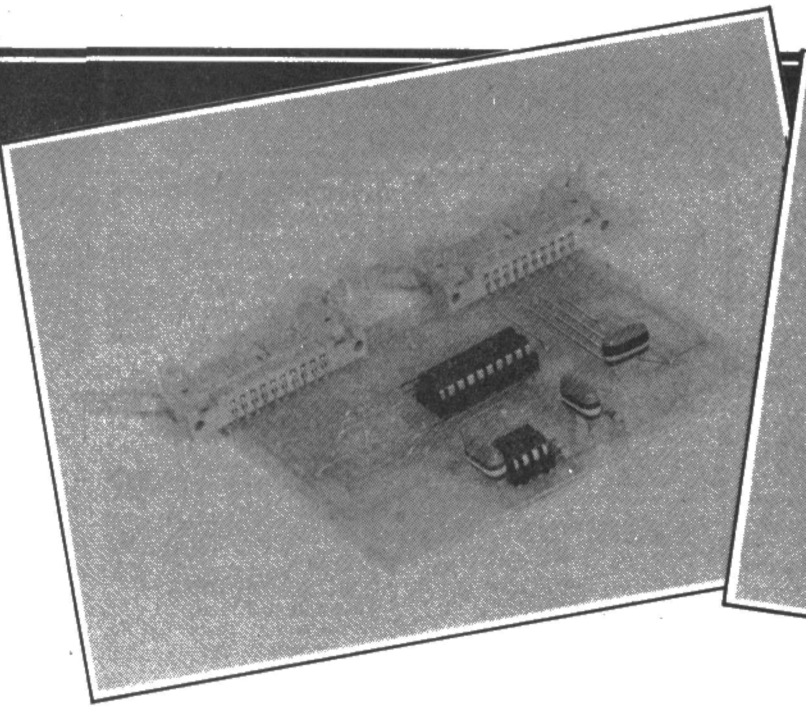
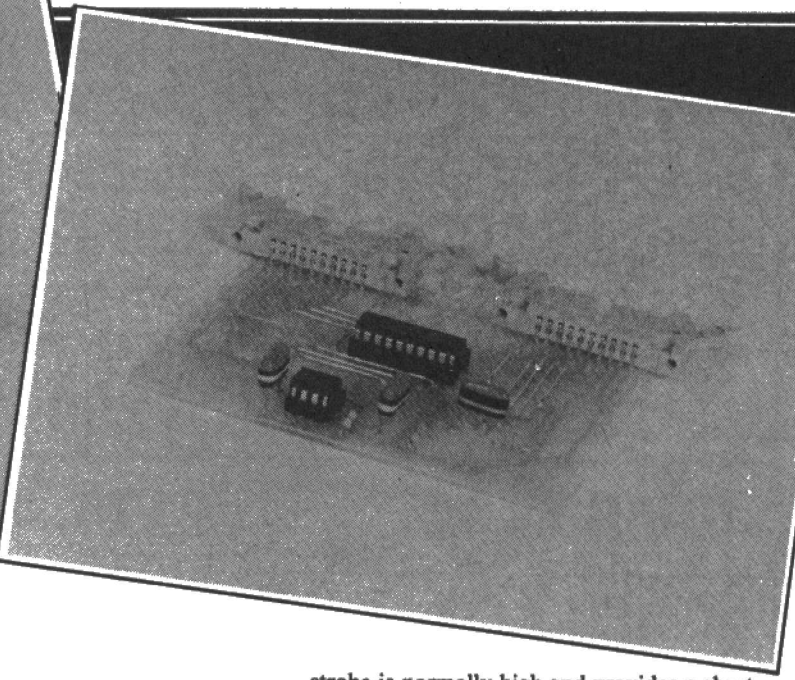
Conclusion

If you keep in mind the way that BASIC statements can be used to bring the assembler into action, either selectively to produce conditional assembly or repetitively to produce macro assembly situations in which the assembler could be used more intelligently, can be recognised. Also the facility to use BASIC variables and expressions within the assembler can be used to further extend the way that programs and macros can be written. There may be other ways in which this clever association between BASIC and the assembler can be exploited so be ready to experiment.

If this article has awakened your interest in the BBC Micro's assembler, you may be interested to know that Mike James's recent book "The BBC Micro: An Expert Guide" published by Granada at £6.95, devotes two chapters to assembly language programming.

E&CM

Watch our for a new series from Mike James starting in our October Issue.

ORIC OUTPUT PORT

John Baker describes an eight bit latched port that is used in conjunction with the Oric's printer port.

The Oric 1 microcomputer has good potential for use in control and measurement applications, with both a parallel printer port and an expansion port that can be used to get data in and out of the machine. The expansion port has the greater flexibility as it provides access to all the data, address, and control bus lines. With the addition of a parallel interface adaptor such as the 6522 a reasonably straightforward but extremely useful 2 by 8 bit (plus handshake lines) input/output port can be produced.

If a simple 8 bit latching output port is all that is required however, the printer port offers the best solution to the problem. Some microcomputers have a built-in PIA such as a 6821 or 6522 which acts as a printer port, but can also act as a latching output port. The printer port of the Oric does not seem to be in this category, with the 8 data lines of this port all carrying a continuous stream of signals even when no output has been directed to the printer. It is therefore necessary to add an external latch.

Timing

Reference to Appendix F on page 151 of the Oric 1 manual (and also reproduced on page 50 of the April 1983 issue of *E& CM*) shows that the 20 way ID connector used for the printer port has the usual strobe, acknowledge, and ten ground connections as well as the eight data lines. The ten ground lines act as screens between adjacent signal lines in addition to providing the earth connections between the computer and printer (or whatev ever).

The strobe line is an output, and this is used to signal to the printer when a byte of information is ready for sending (a process which is sometimes irreverently, but graphically, called "grabbing a byte"). The strobe is normally high and provides a short negative pulse when an output byte is present on the data lines. The acknowledge terminal is an input and is normally high. The printer or other output device must supply a negative strobe to this once it has successfully received a byte of information – the computer will not send any more information until it has received this signal. When a printer is being used this serves to ensure that the computer cannot send data at a rate which is too fast for the printer, however when the computer is being used to control electronic equipment, there would not normally be any need for constraints on the rate of data flow. In this case the acknowledge input is not of great importance, although it must still be supplied with a suitable signal or the flow of data will be halted!

Figure 1 shows the waveforms on the signal lines as data flows from the computer (only two data lines are shown).

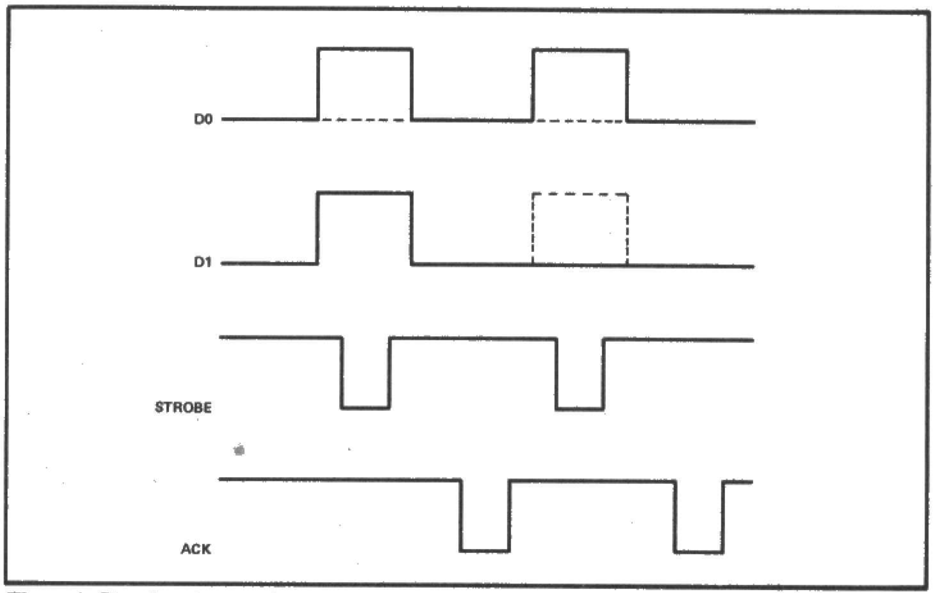
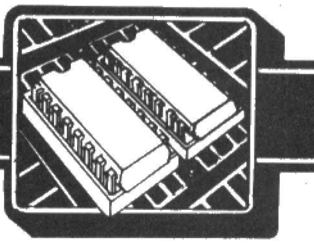


Figure 1. Showing the waveforms on the signal lines as data is transferred from the computer to an output device.



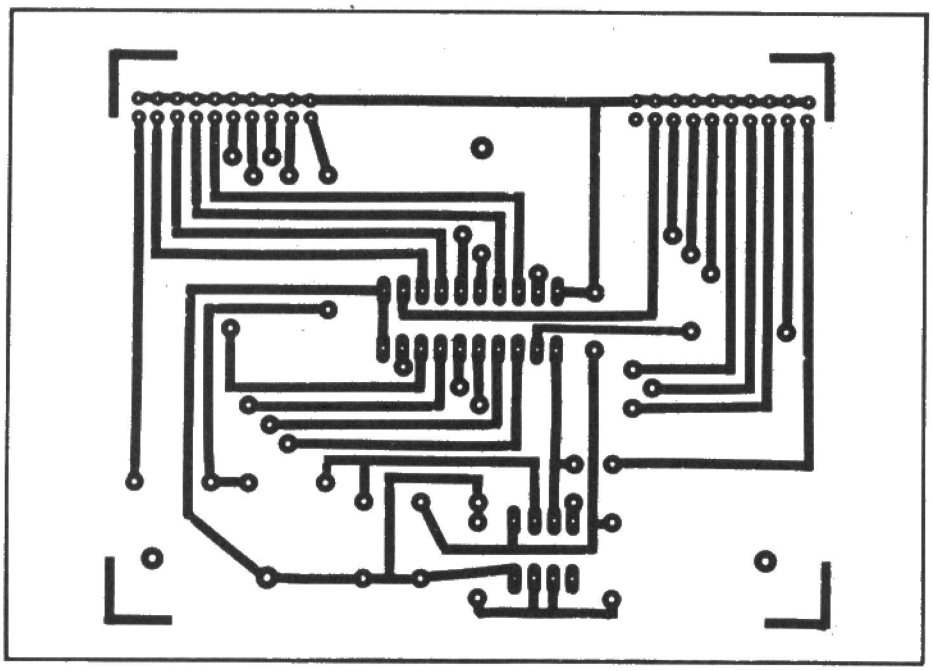


Figure 2. The Oric Output Port's PCB foil pattern (shown full size).

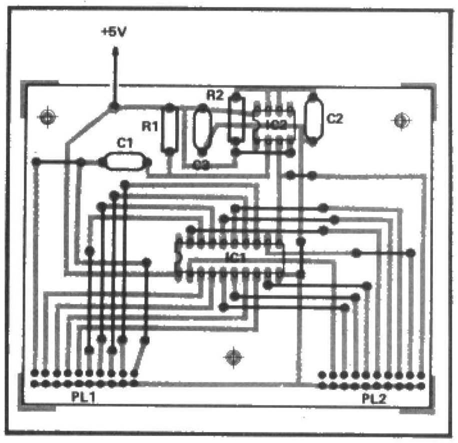


Figure 3. The overlay showing the positioning of components on the Output Port's PCB.

The Circuit

The full circuit diagram of the Oric Output Port is shown in **Fig 4.**

IC1 is a 74LS273 octal D type flip/flop, and this is used to latch the data lines. With the NMR input (pin 1) tied to the positive supply rail the outputs latch at whatever state is present on the data inputs as the CP input (pin 11) goes through a high-to-low transition. On the face of it, simply applying the signal from the strobe output to the CP input of IC1 should provide the latching action at the appropriate time, but in practice this system does not seem to provide reliable results.

Buffering the strobe signal did not seem to improve matters, and the system finally adopted was to use a 555 timer IC in the monostable mode to provide the latching pulse. C1 and R1 are used to shape the strobe pulse to give a very brief negative trigger pulse for IC2. IC2 then provides a positive output pulse of about 50ms in duration, with R2 and C2 setting this pulse length. IC1 is therefore latched about 50ms after the start of the strobe pulse. The signal from IC2 is available on the output connector (PL2).

A circuit to produce an acknowledge signal could be included in the unit, but an alternative which seems to be very reliable is simply to connect the strobe and acknowledge lines together. It can be seen from **Fig 1** that the strobe pulse is short and negative, much as is required at the acknowledge input. Although using the strobe pulse in this way gives a slightly premature acknowledge signal this does not seem to be of any practical significance.

The circuit requires a 5 volt supply and has a current consumption of about 25 milliamps. If no other convenient power source is available the 5 volt output at pin 33 of the Oric's expansion port can be utilised.

Construction

The printed circuit layout for the unit is given in **Fig 2.** PL1 and PL2 are both standard 20 way, right angle, printed circuit mounting, ID connectors. A 20 way double header ribbon cable is used to connect the unit to the Oric's printer port. If the 5V supply is obtained from the expansion port of the Oric, a 34 way header socket will be needed, (refer to page 151 of the Oric i Manual). Most ID connectors are polarised and cannot be connected the wrong way round, but be careful to connect everything correctly if you are using a type which is not polarised.

In Use

Data can be sent to the printer port (and the latch circuit) using the command:-

LPRINT CHRS(X)

Here "X" can simply be the number to be sent to the port (i.e. LPRINT CHRS(255)), or a numeric variable can be used. This gives good versatility and makes the port easy to use. The number is fed to the output port in binary form, with the most significant (left hand) bit representing data line D7, through to the least significant (right hand) bit which corresponds to data line D0.

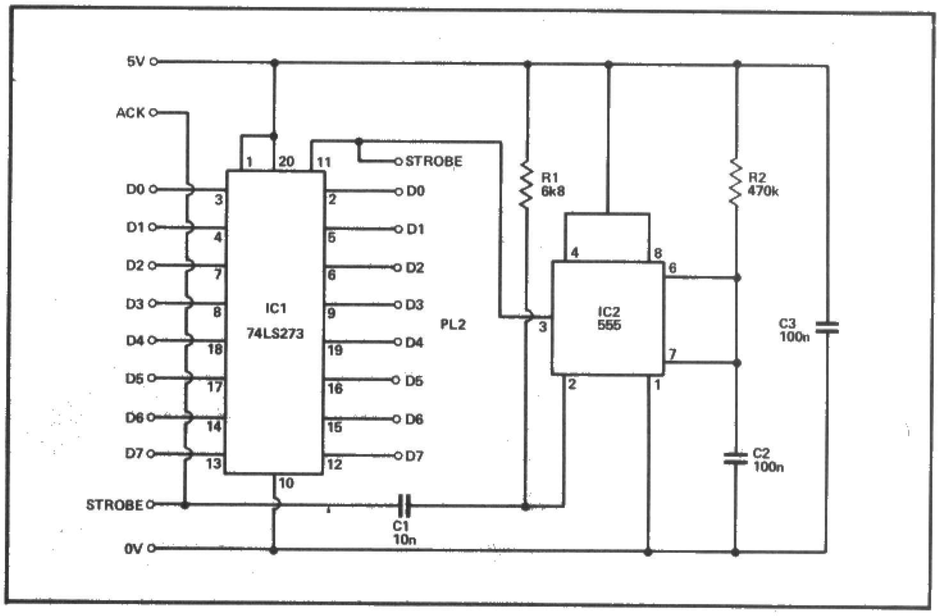
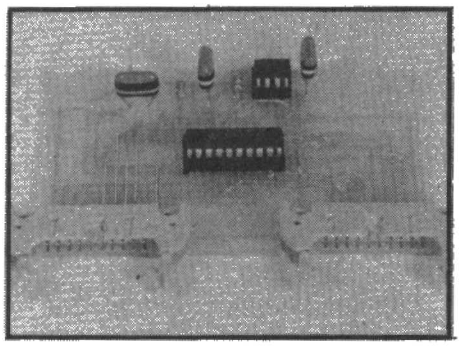


Figure 4. Full circuit diagram of the Oric Output Port.


The completed Output Port PCB.

There is effectively individual control over each bit of the output port since each bit can be switched on or off by adding to or subtracting from the number sent to the output port the number represented by the bit concerned. For example, data line D3 represents 8 (in decimal) when it is in the high state. If (say) 82 was the number currently at the output port (which is 01010010 in binary), D3 would be in the low state. Adding 8 to the number sent to the output port would give 90, or 01011010 in binary, and D3 would go high. Subtracting 8 from the number sent to the output port would take things back to their original condition with D3 low once again. Provided a 1 is not written in error to an output that is already in the high state, and the correct number is used, only the one bit will be affected.

This may seem to be a trivial point, but it is important in applications where the port is being used to control a number of separate pieces of equipment. This is also one of the reasons that it is important to be able to send a numeric variable to the output port (e.g. X=X+8, LPRINT CHRS(X) can be used to set D3 from low to high regardless of the state of other outputs).

The 8 data outputs of the port are at 74LS logic levels and there should be no difficulty in interfacing the unit with relay drivers, opto-isolators, digital to analogue converters, etc., and details of suitable circuits have been given in past issues of *E&CM.*

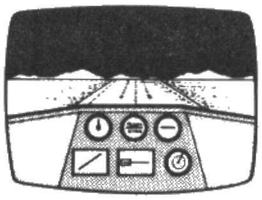
E&CM

PARTS LIST

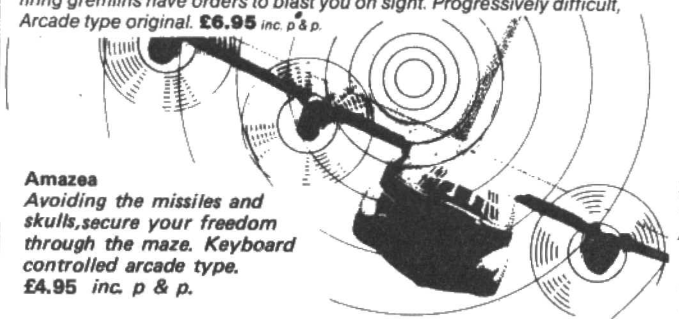
Resistors (¼W, 5%)	
R1	6k8
R2	470k
Capacitors	
C1	10n
C2,3	100n
Semiconductors	
IC1	74LS273
IC2	555
Connectors	
PL1,2	20 way IDC plug with right angle pins
Miscellaneous	
PCB, DIL sockets, connecting cable, etc.	

NEW! ORIC 1

Flight Simulator
With airspeed, radar, ground/vertical flight path indicators, artificial horizon and altimeter with continuous flight monitor on engine condition, stall speed etc., full colour, Hi-res graphics, 3D and sound. **£6.95** inc. p & p.



Moonster
Defeat the guardians of Hyper to retrieve the life crystal. Mean zappo firing gremlins have orders to blast you on sight. Progressively difficult, Arcade type original. **£6.95** inc. p & p.



Amazea
Avoiding the missiles and skulls, secure your freedom through the maze. Keyboard controlled arcade type. £4.95 inc. p & p.

Space Crystal
Space trading adventure game. Fascinating 3D space travel on your voyage for the Illudium crystal. Black holes, hyper-drive, gamble planets and space pirates. Full colour and sound. £6.95 inc. p & p.

Romany Oracle
Your own private gypsy, this programme uses exactly the established method of card fortune telling and gives present and future readings from interpretation of the cards and combinations of them. £6.95 inc. p & p.

Original programmes required, cash or percentage terms.

DRAGON 32 Cassettes

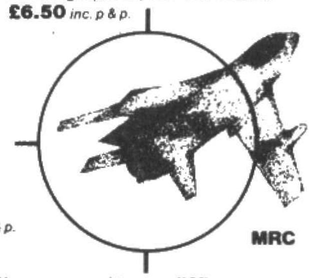
MRC
Air combat and ground strike, "joystick" controlled sights cannon fire and bomb release. Full colour, Hi-res graphics, 3D and sound. **£6.50** inc. p & p.

Fruit
Compulsive, addictive, fruit machine. Using your skill with the holds, turn your £5 stake into £10 and beat the system. Hi-res graphics and sound. **£4.95** inc. p & p.

Look and Learn
Educational picture recognition. Hi-res graphics, text and colour. Age 3–10yrs. **£4.95** inc. p & p.

Pontoon
Super Hi-res graphics, full colour definition and card display. Fascinating, absorbing. **£4.95** inc. p & p.

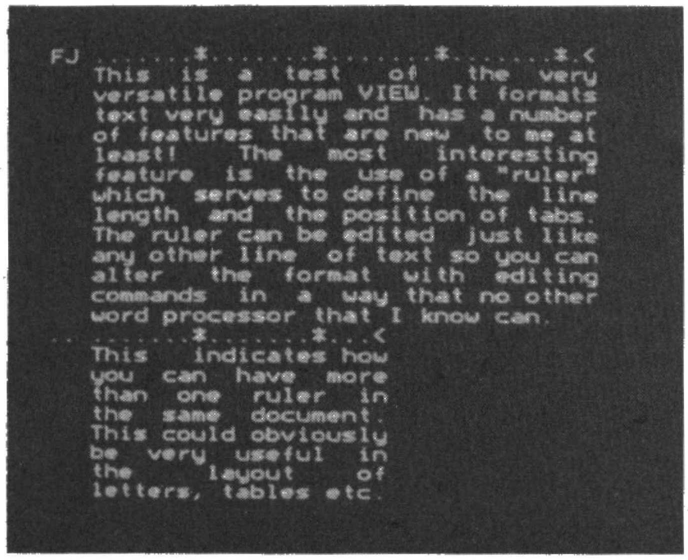
Tango Foxtrot 01
Using your skill, locate the airstrip with the radar scanner, fly your aircraft through hazardous crosswinds and air pockets to land safely. Watch your fuel; watch your artificial horizon; be sure you are level on final approach. Full colour, Hi-res graphics, 3D and sound. **£6.50** inc. p & p.



Missile Command
City defense from attacking missiles, progressively difficult. Arcade type. £4.95 inc. p & p.

Graphics Demonstrator
3D rotation, drawing with joystick, approx. 50 examples of Dragon's impressive graphics. £4.95 inc. p & p.

QUARK DATA P.O. BOX 61, Swindon, Wilts. Tel. (0793) 40661.
Please supply.......................................
....
....
Cheque for total amount enclosed ....
Name.......................................
Address.......................................
....



```
FJ .......*.......*.......*.......*.<
   This  is  a  test   of   the  very
   versatile program VIEW. It formats
   text very easily and  has a number
   of features that are new  to me at
   least!    The   most   interesting
   feature  is  the  use of a "ruler"
   which  serves to define  the  line
   length  and  the position of tabs.
   The ruler can be edited  just like
   any other line  of text so you can
   alter   the   format   with  editing
   commands  in  a  way that no other
   word processor that I know can.
.......*.......*...<
   This   indicates how
   you  can  have  more
   than  one  ruler  in
   the  same  document.
   This could obviously
   be  very  useful  in
   the      layout    of
   letters, tables etc.
```

VIEW REVIEWED

Last month S. M. Gee looked at the Wordwise word processing package for the BBC micro. This month Acornsoft's View is investigated.

Although ACORNSOFT announced VIEW at an early stage it only became available this Spring. In keeping with all ACORNSOFT products it is impressively packaged and very well documented. It comes with two booklets, an introductory volume, "Into VIEW", which presents an overview of all the program's facilities and a spiral bound "Guide" which acts as a reference to VIEW's commands, their use and their effects.

VIEW is a more sophisticated package than Wordwise (see last month) and this means that in the long run it offers extra facilities, such as the ability to cope with form letters. To counter this sophistication however, the user experiences rather more difficulty in getting started.

To get into VIEW, once it has been installed, the user must type *WORD. This puts View into the "command" mode and the screen displays the information that there is no text in memory, that the package is currently editing No File, that the computer is in Mode 7 and that the printer default is operating. The 'invitation to type' prompt in this mode is => but this is for commands to do with saving and loading files or printing out. Text is entered, naturally enough, in "text" mode and, as with WORDWISE, the ESCAPE key is used to switch between the two modes. to start initially however, NEW (return) must be typed before pressing ESCAPE.

Handy Rules

One of the most important features of VIEW, and one which is very different from any other word processing program I personally have ever used, is apparent as soon as the text mode is entered. This is the "ruler" which appears at the top of the page as a row of dots, interspersed with asterisks and with a < to indicate the right margin. The dots represent character spaces and the asterisks are TAB stops. Rather than using embedded commands to determine line width, the user sets up his own ruler, not only for line width but also for the tab stops, if any are required. This certainly makes tabs more convenient, even for simple applications such as paragraph indents, and is, in my experience, a good system for coping with complicated tabulations. A method of redefining another character, for example the space bar, to have the function of the TAB key is a little finishing touch that I was impressed with. Having struggled in the past to produce neat tables on a word processor, I welcome these facilities in VIEW.

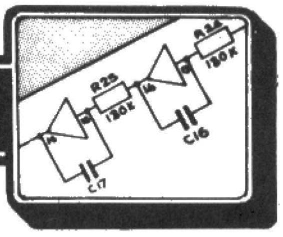
Up to 128 rulers can be used in a document and existing rulers can be altered, new ones made up, or use made of the "default" ruler, which appears at the top of the screen at switch on. A ruler may be up to 132 characters wide but it is important to bear in mind the width of the paper the document is to be printed on. If a ruler is changed having already typed in some text below it, the text can very easily be reformed. This facility enables plenty of experiment with layout. Unlike WORDWISE, with VIEW what is seen on the screen as text is typed in, is very close to what will eventually be seen on paper. The screen acts as a "window" onto the whole document so, if a line width of more characters than the screen is chosen, only part of the text will be seen at a time. This would be a serious restriction if VIEW, like WORDWISE, only operated in Mode 7. However, VIEW can be used in any of the BBC Micro's modes (to switch mode type Mode (number) RETURN while in command mode) and Mode 3 is particularly useful in that it has eighty columns across the screen. its default ruler has 74 characters which is convenient for A4 size paper and for the nearly equivalent 9.5" by 11". There is one important consideration about using Mode 3. it really requires a high quality black and white monitor. If, as in my case, a small TV set is used, then Mode 3 produces not only almost illegible display but is also tiring on the eyes, and even distrubing.

Stored Commands

There are two types of commands used in text mode, "stored" commands, which correspond to WORDWISE's embedded commands, and "immediate" commands. The first group of immediate commands encountered in the Guide are about moving the cursor. Again, the arrow keys are used, but in the case of VIEW they can only be used alone or with SHIFT, which makes the cursor scan between words horizontally or jump 24 lines at a time up or down. To get to the top or bottom of the page or the extreme left or right of a line the appropriate function keys are used. VIEW relies very much more on the function keys than WORDWISE and all but one of them have three separate functions, according to whether they are pressed alone, together with the SHIFT key or with CTRL. This may sound complicated, but in practice it is easy to get used to and the prompt card that is kept underneath the clear plastic strip is well laid out. It does mean that it is not possible to define the function keys for other uses, which is allowed in WORDWISE for using a function key plus SHIFT. This does not seem a very useful feature of WORDWISE, however, though VIEW's range of pre-defined facilities seems to be pretty comprehensive. There are commands for inserting both single characters and sections of new text, for deleting characters and sections of text and for swapping case. To replace characters, the cursor is moved to the appropriate place and the new text typed over the old. In other words, unless instructed to do otherwise, VIEW overwrites rather than inserts – until it reaches the end of the line it is on, when it then continues to insert on the next line.

The way VIEW treats a line is crucially different from the way WORDWISE does. In WORDWISE the carriage return acts like any other character which means that it can be deleted. This is not so in VIEW and a line once created, either by the user pressing RETURN or by the program supplying one when the right margin is reached, is treated as a special unit. So although it is possible to move the cursor to the left and right within a line, and to replace characters on the line, when the end of the line is reached there is no automatic moving on to the next. The special commands, format, concatenate and split lines are provided to cope with certain problems that this creates.

There are immediate commands for setting markers and moving and deleting blocks of text using them. To copy text, VIEW uses the COPY key at the bottom right of the keyboard – the prompt card can be searched in vain for this facility.



Turning now to the stored commands, these are used for page layout, with facilities for line spacing, justification, page and line length, centering lines, page numbering, margins and for headers and footers, to name but a selection. The extra facility that makes VIEW's stored commands really powerful is the ability to put them together in "macros". It is possible to define a macro that contains not only formatting commands but also parameters and lines of text. The macro facility, therefore, is the one that can be used for sending out standard letters and other frequently repeated documents that require customising each time they are used.

Search And Replace

VIEW also offers a more powerful way of searching through text and replacing strings of letters than WORDWISE. I particularly liked the "wild search" facility which allows a ? to be substituted in place of letters. This is very useful for looking for misspellings of words. Once SEARCH has located the first occurrence of the word of interest, VIEW can find the next one by pressing the NEXT MATCH function key. This is very useful. I also liked the way in which upper and lower case letters in the original are taken account of when using CHANGE, the command that replaces every occurrence of a word in the original by the alternative you supply. The command REPLACE allows the user to make a separate decision about substitution at each occurrence by pressing Y or N.

WILD, SEARCH, CHANGE and REPLACE are all command mode commands. Other of these commands allow files and parts of files to be saved or loaded, and to count words and print material out.

One notable advantage that VIEW appears to have over WORDWISE, is the ability to edit texts, using the command EDIT, which are too long to fit in memory all at once. In this respect VIEW acts as a disk to disk editor. As in Mode 7, with VIEW installed, the BBC Micro cannot hold many more than 3000 words (many fewer in Mode 3) this is very important.

Some of VIEW's advanced features can only be used in conjunction with other sophisticated equipment. for example, highlighting or monospacing require a printer that will support them. The latter also needs a special printer driver, which is available from Acornsoft on cassette. Other advanced features, such as its number registers, are available to every user.

WORDWISE vs VIEW

Having tried out these two word processors I am very impressed with both of them and find them both very good value for money. For £39 WORDWISE offers an extremely user friendly program that is sufficiently powerful for most everyday applications. Because it does not offer a straightforward way of handling large files, however, it does have limitations for a user like myself – for example, using WORDWISE I would have had no option other than splitting this article into two separate files. Its attraction is, however, that it is very easy to get started with – the complete antedote to WORDSTAR – and would make an excellent choice for someone who was rather apprehensive about word processing. The cassette tape that is supplied with it, containing a sample document – in fact part of the manual – is a nice reassuring extra, as it gives the newcomer a chance to see the relationship between what is displayed on the screen and what is in print before they even start.

Although more difficult to get the hang of, VIEW is undoubtedly the more powerful program and I feel that it is probably worth making the extra initial effort required because of the advantages it offers. To my mind, the most important of these are disk to disk editing and versatility in drawing up tabulations. At £52 it is certainly good value for money as it makes the BBC Micro as efficient a text handler as any computer of comparable price.

E&CM

E&CM PCB SERVICE

This month we introduce a new service to our readers – the *E&CM* PCB service. Each month we'll be producing high quality boards to support the projects featured in *E&CM*. The prices shown include VAT but a 45p post and packing charge must be added to the cost of the board(s) ordered.

April 1983

TV to RGB Conversion	**£2.70**

July 1983

Power Control For Micros

Relay Board	**£2.02**
DAC Board	**£1.77**
Stepper Motor Driver	**£1.59**
BBC Sequencer Interface	**£2.10**

August 1983

Oric Output Port	**£2.10**
Spectrum Sound Board	**£2.20**

HOW TO ORDER

List the boards required and add 45p post and packing charge to the total cost of the boards. Send your order with a cheque or postal order to:

ECM PCB Service, 155 Farringdon Road, London EC1R 3AD.

Please supply the following PCBs:

....

....

Post & Packing 45p

TOTAL £ ______

Signed .... Date ....

Name .... (please print)

Address ....

....

....

THE COMPUTER BRAIN

Mike James ponders the question as to whether a program is intelligent or just clever. He also describes a program that learns to play a game.

One of the fascinating things about AI is the way that the same method, or program, can be described, or thought of, in more than one way. This fascination can be something of a problem in that it is often difficult to work out what is really new! It is possible to look at a program that solves some problem and not realise that it is far from new simply because it is explained in terms that are unfamiliar. Perhaps the single most important theme of Computer Brain has been that there is nothing very different about intelligent programs and that they can be understood without any difficult theory. In this final part of the Computer Brain we look back at some of the central ideas of AI and try to show how they fit together. to round off the series we return to a problem that was introduced earlier – how to play noughts and crosses. The new element is that this noughts and crosses program learns how to play from a human teacher.

Intelligent or just clever?

There is a very thin dividing line between clever programming and artificial intelligence. Indeed, it is possible that there is no such thing as an intelligent program – just clever programs that become increasingly clever. For example, Part 6 of Computer Brain (January 1983) presented a fairly short BASIC program that would carry on a limited conversation. The limitations were partly due to the fact that it was written in BASIC but were mainly due to the rather crude method that lay behind the program. By detecting the presence of certain keywords within sentences the program could offer relevant, if not meaningful, sentences in return. For example, if you used a sentence with the word "WHY" in it, the program would answer "Some questions are difficult to answer", a vague, enigmatic response which serves in most situations! If you want to try a small experiment just say "Some questions are difficult to answer" in response to anything said to you that contains the word "WHY". To improve on the program also smile of frown depending on whether or not you judge the situation to be light hearted or serious! Apart from being a little boring you will find that this automatic response is accepted by a large number of people.

The point is that by applying some very simple rules you can give the impression of intelligence. Human beings are, intelligent and if you can raise their expectations to a high enough level, they will try to interpret almost anything as evidence of intelligence in someone or something else. In the case of computers, the public's expectations have been raised almost to the roof and it is fairly easy to convince an innocent onlooker that some trivial program or other is the cleverest thing in the universe.

Machines quickly become endowed not only with human intelligence but also with human personality and this is nothing new. In the days of steam engines huge hunks of brass and steel were often referred to as "she" or even by a first name! Humans are very, very willing to lend some of their intelligence and personality to the simplest of machines. The computer dialogue program described earlier first saw the light of day as "ELIZA", an exercise in artificial intelligence. Computers were so new at the time that a considerable number of people were taken in by its performance and had deep and meaningful conversations with it. So seriously was it taken that psychiatrists wanted to use it with real patients and patients even preferred it! If you think that we have now reached the point of much greater sophistication that simple ELIZA-like programs would be old hat you would be disappointed. A recent BBC Horizon program featured the original ELIZA both as a historical event and as an example of what computers can do in areas more usually thought of as human territory. The final image of a secretary in deep conversation with what readers of Computer Brain now know to be a simple program must have left an impression that computers were already much better than any simple human.

This willingness to believe in the intelligence of computers has two important aspects. Firstly it means that we can achieve some useful results without too much effort by borrowing some of the user's intelligence and secondly we must ourselves beware of becoming believers too easily!

The Turing Test

The traditional way of guarding against being taken in and thinking that a clever program has achieved intelligence is the so called Turing Test – named after Alan Turing who first proposed it. The basic idea is that any program that claims intelligence should be tested against a human, the only certain form of intelligence that we know of. This test consists of placing the program and the human in a box and allowing other humans to communicate with both over a VDU or some other common means of communication. If the external observers cannot tell with any certainty which is the computer program and which is the human then for all practical purposes the program may be called intelligent. This simple test has many problems, for example, should the human make an active effort to conceal or reveal the truth, but it is appealing as a description of what we should expect from an intelligent program.

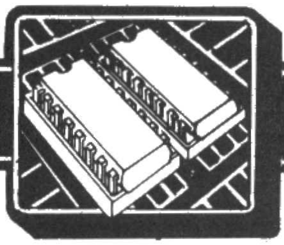
The main trouble is that the Turing test is a 'black box' test in that the observers don't worry themselves about how things are working inside the boxes, only that the outputs are appropriate. The reason why this is a problem is that it is possible that a program will pass the Turing test and receive the label 'intelligent' but to the computer scientists that created it, it seems rather more simple in its workings than true human intelligence. Some programs can already pass limited forms of the Turing test where the questions that the observers can put to the program are restricted to a narrow range of topics. For example, chess programs are now so good that it takes a master to tell the difference between a human player and a program. In this sense a chess program is intelligent when the Turing test is restricted to questions about games of chess but as we know the program uses methods that are unlikely to be the same ones that humans use. A chess program searches a move tree, uses exact evaluation functions, applies heuristic rules and remembers openings and endgames more accurately than any human ever could. In other words, much of its performance comes from clever programming combined with the raw speed of calculation offered by a computer.

Is AI Different?

There is a point of view that says that AI is not really different from normal programming and computer science. This is certainly true in that so far we have failed to write any programs that can claim to be intelligent. Any AI program written to date uses nothing so difficult that it wouldn't be recognised and understood by a good programmer. The trouble is that it is difficult to see how you can write a program that you can claim is intelligent when you also know in great detail how it works! In this sense AI is a subject chasing a moving point. Every time we manage to write a program that does some task that we thought required intelligence, we know exactly how it works and so the task becomes demoted to the ranks of automation. An intelligent program would have to be such that it fooled not only its audience, like the ELIZA program, but also fooled its creators and this is something that is difficult to imagine. ▶

Listing of the program that can learn to play noughts and crosses.

```
  10 REM LEARNING TO PLAY
  20 REM NOUGHTS AND CROSSES
  30 GOSUB 1000
  40 L=0:M=0
  50 GOSUB 2000
  55 M=M+1
  60 GOSUB 3000
  65 GOSUB 2000
  66 GOSUB 8500
  70 IF L<>0  THEN GOTO 9000
  80 GOSUB 4000
  85 GOSUB 2000
  90 GOSUB 8500
  95 M=M+1
 100 IF L<>0 THEN GOTO 9000
 110 GOTO 60

1000 DIM M(100),R(100)
1010 DIM B(3,3)
1020 N=100
1030 RETURN

2000 FOR I=1 TO 3
2010 FOR J=1 TO 3
2020 IF B(I,J)=0 THEN PRINT ".";
2030 IF B(I,J)=1 THEN PRINT "X";
2040 IF B(I,J)=2 THEN PRINT "O";
2060 NEXT J
2070 PRINT
2080 NEXT I
2090 RETURN

3000 PRINT
3010 PRINT "YOUR MOVE (ROW,COL)";
3020 INPUT R,C
3030 IF R>3 OR R<1 THEN GOTO 3010
3040 IF C>3 OR C<1 THEN GOTO 3010
3050 IF B(R,C)=0 THEN GOTO 3080
3060 PRINT "ALREADY OCCUPIED !"
3070 GOTO 3010
3080 B(R,C)=1
3090 RETURN

4000 PRINT
4005 GOSUB 5000
4010 GOSUB 6000
4030 IF S>N THEN GOTO 4500
4040 IF F=0 THEN GOSUB 6500
4050 IF F=1 THEN GOSUB 7000
4060 RETURN

4500 PRINT "I HAVE RUN OUT OF MEMORY"
4510 STOP

5000 P=0
5010 FOR I=1 TO 3
5020 FOR J=1 TO 3
5030 P=P*10+B(I,J)
5040 NEXT J
5050 NEXT I
5060 RETURN

6000 S=1
6010 F=0
6020 IF S>N THEN RETURN
6030 IF M(S)=0 THEN RETURN
6040 IF M(S)=P THEN F=1:RETURN
6050 S=S+1
6060 GOTO 6020

6500 M(S)=P
6510 Q=P
6520 R(S)=P
6530 GOSUB 8000
6540 RETURN

7000 Q=R(S)
7010 GOSUB 8000
7020 RETURN

7500 Q=R(S)
7510 X=0:Y=0
7520 A=Q-INT(Q/10)*10
7530 Q=INT(Q/10)
7540 X=X+1
7550 IF A<>0 THEN GOTO 7520
7560 Y=3
7570 IF X=1 THEN GOTO 7610
7580 FOR I=1 TO X-1
7590 Y=Y*10
7600 NEXT I
7610 R(S)=R(S)+Y
7620 RETURN

8000 X=0
8010 A=Q-INT(Q/10)*10
8020 Q=INT(Q/10)
8030 X=X+1
8040 IF A=0 THEN GOTO 8100
8050 IF X=9 THEN L=1:RETURN
8060 GOTO 8010
8100 X=9-X
8110 R=INT(X/3)
8120 C=X-R*3
8130 B(R+1,C+1)=2
8140 RETURN

8500 FOR I=1 TO 3
8510 IF B(I,1)=1 AND B(I,2)=1 AND B(I,3)=1 THEN L=1
8520 IF B(I,1)=2 AND B(I,2)=2 AND B(I,3)=2 THEN L=2
8530 IF B(1,I)=1 AND B(2,I)=1 AND B(3,I)=1 THEN L=1
8540 IF B(1,I)=2 AND B(2,I)=2 AND B(3,I)=2 THEN L=2
8550 NEXT I
8560 IF B(1,1)=1 AND B(2,2)=1 AND B(3,3)=1 THEN L=1
8570 IF B(1,1)=2 AND B(2,2)=2 AND B(3,3)=2 THEN L=2
8580 IF B(1,3)=1 AND B(2,2)=1 AND B(3,1)=1 THEN L=1
8590 IF B(1,3)=2 AND B(2,2)=2 AND B(3,1)=2 THEN L=2
8600 RETURN

9000 IF L<>1 THEN GOTO 9050
9010 PRINT "YOU WIN..."
9020 PRINT "BUT I WILL LEARN FROM MY MISTAKE!"
9040 GOSUB 7500
9050 IF M=9 THEN GOTO 9100
9060 PRINT "I WIN"
9100 PRINT "ANOTHER GAME ";
9110 INPUT A$
9120 IF LEFT$(A$,1)="N" THEN STOP
9130 FOR I=1 TO 3
9140 FOR J=1 TO 3
9150 B(I,J)=0
9160 NEXT J
9170 NEXT I
9180 GOTO 40
```



Even though there isn't the magic of real machine intelligence to mark AI out from other areas of programming, there are a collection of methods that it can call its own. The idea of an algorithm loosely applied as a heuristic is something that characterises AI programs. The tree structure seems to crop up so often in a wide range of apparently different problems that it deserves to be called the fundamental data structure of AI. Indeed you could say that the tree is to AI what the array is to mathematical programming. Much AI theory comes from other disciplines – logic, philosophy, probability etc. – but these appear with lesser frequency. Even so, the AI programmer needs to be something of a polymath.

It is worth trying to see how many of these ideas can be found in this month's final example which returns to the game of noughts and crosses. In particular, try to see its relation to the description given earlier in the series (November, 1982) of the game in terms of a move tree.

Learning Noughts And Crosses

Noughts and crosses is a useful game to use to illustrate AI methods because it is easy enough to make the programs sufficiently short to be written in BASIC, but just difficult enough to make a good demonstration. In the last two month's episodes of Computer Brain, the emphasis has been on the application of IF < condition > . . . THEN < conclusion > type rules in expert systems. In earlier articles (February and March) we looked at ways that programs can learn from their experiences. It would be interesting to see how the two ideas could be combined together to give a program that learns a set of IF . . THEN . . rules to solve some problem. The difficult part is selecting a problem that is easy enough to make the resulting program understandable and this is where noughts and crosses makes its entry once again.

To be able to learn anything the first requirement is a good memory. The reason for this is that to learn it must be possible to remember what worked and what didn't. In the case of noughts and crosses its necessary to remember moves that lost the game and then try not to play them ever again. A noughts and crosses board consists of nine playing positions each of which can either hold a blank, an X or a 0. If we code blank as zero, the X as 1 and the 0 as 2 then any position in a game of noughts and crosses can be recorded as a sequence of digits consisting of only 0, 1 or 2. For example, 0000000010 would correspond to a board with eight blank positions and one X. It would be quite possible to record board positions by saving the digits as strings but in an effort to produce a program that will work on a wide variety of machines it is better to save the digits as numbers in an array. This method will only work with versions of BASIC that can store nine digit numbers. To find out if your version of BASIC can do this, try –

```
10 A=333333333
20 PRINT A
```

If the result printed is exactly the same as the number stored in A by line 10 then your BASIC can work with nine digit numbers. If this is not the case then you can either try to make use of integer variables (if your BASIC has these) or you can re-write the program to use strings.

The program given below uses this idea of saving board positions as nine digit numbers in the array M. Subroutine 5000 will convert the current position held in the two-dimensional array B into a single number stored in P. The program begins by offering the human player a move (subroutine 3000) and then recording in the array M the resulting board position. This is carried out by subroutine 4000, which calls subroutine 5000 to code the board position into a single number P, then check to see if the board position is already stored in M using subroutine 6000 and it it isn't it, subroutine 6500 is called to store it.

So far we have a program that remembers new board positions as a game is played. But how does it choose its own move and how does it learn? The next stage in developing the program is to relaise that the response to the human player's move simply depends on the current state of the board. This means that it is possible to think of playing noughts and crosses as –

IF current position is X THEN make move Y

In other words, as another application of the IF . . THEN rules of expert systems! Along with the array M there is a corresponding array R which is used to store responses. The way that this works is that if the current board position is stored in M(I) then the response is stored in R(I). The best way to store the response is using the same code as for the current board position. As the program hasn't learned anthing about noughts and crosses as yet its response might as well be to the first free board location. This is indeed what happens. As each new board position is encountered subroutine 4000 saves it in M and in R and then calls subroutine 8000 to make a move into the first free space.

Obviously, almost random play of this sort is easy to beat and so the program remembers the details of each board position and what it played as a response but always loses! The next stage is to weed out bad IF . . THEN rules. This is surprisingly easy to do. Let's suppose that the program has just lost a game then the move that it made just before it lost was a very bad move. The obvious thing to do is to search through memory to find the IF . . THEN rule that lost the game and while leaving the IF . . part alone change the response to the current board position that follows the THEN. As moves are made into the first free space, and this is marked in the nine digit number stored in R by 0, all we have to do to stop this being the response is to store a digit other than zero, 3 say, in this position. This is what subroutine 7500 does. Next time that the losing board position appears the program will play a different move as a response and if this once again loses the game it too will be 'blanked out' with a 3. This process of changing the move will continue until the correct, i.e. non-losing response, is found.

If the program listed is run it is very easy to beat at first but slowly it will learn how not to lose. And in noughts and crosses knowing how not to lose is almost as good as knowing how to win! You should be able to work out the rest of the program for yourself but to help the subroutine structure is –

```
  10 main program
1000 initialisation
2000 print board
3000 human's move
4000 computer's move (uses other subroutines)
5000 code board position into a single number
6000 find current position in M
6500 position not found, store in M and R
7000 position found get response from R
7500 lost game so remove first 0 in losing move
8000 find first free position for computer's move
8500 has anyone won yet?
9000 end of game
```

There is plenty of work left to do on this program. In particular there is a one game that it will never learn to avoid losing. What is it? Why is it impossible for the program to avoid losing it no matter how much it learns? What has this got to do with the game tree? What simple modification to the program will allow it to avoid losing? All these questions are left for you to answer but if you have followed this series none of them should prove problematic.

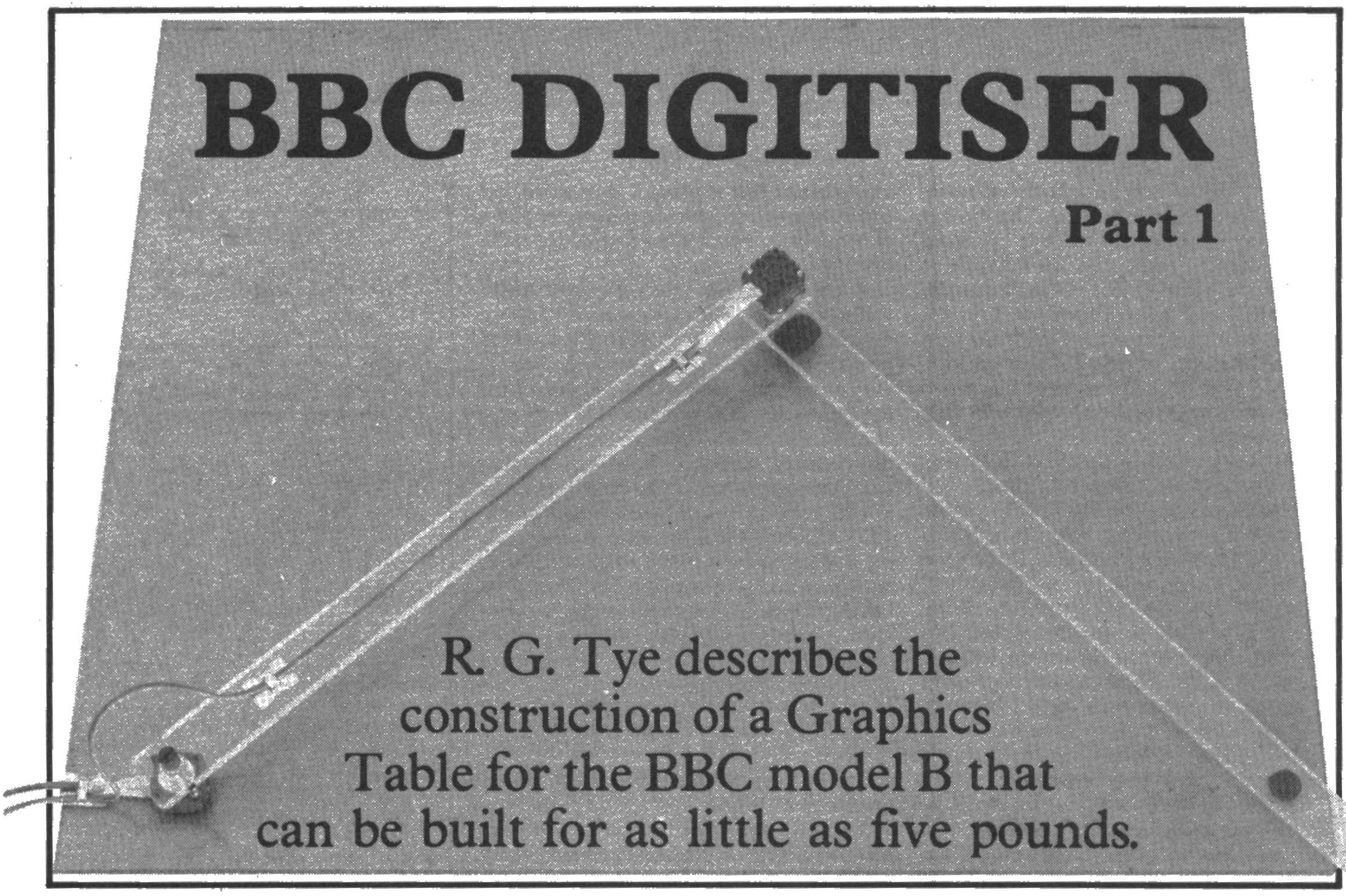
Watch out for a new series from Mike James starting in our

E&CM

BBC DIGITISER

Part 1

R. G. Tye describes the construction of a Graphics Table for the BBC model B that can be built for as little as five pounds.



Commercial digitisers for the BBC micro are freely available but can vary in price from 20 to over 100 pounds. Adopting a do it yourself policy can save money as the project described here offers a precision digitiser at a fraction of the cost of its commercial equivalents.

The digitiser consists of two arms, one pivoted at the edge of the drawing board, and the second printed at the end of the first. Now by doing some trigonometry and knowing the length of the arms, it is possible to calculate the position of the first arm relative to its pivot providing the angle between the arm and the edge of the baseboard is known. Similarly, if the angle of the second arm to the first arm is given, by calculating back, the position of the pointer at the end of the second arm can be found. In order to measure the angles, ordinary linear tracked potentiometers connected to the A/D part on the micro can be used. The potentiometers can also serve as pivots for the arms. **Fig 3** shows the general layout of the digitiser.

Now doing a series of complex calculations very quickly is just the sort of task computers were built for, and the BBC micro romps away with this problem; in fact, delays are built into the program to slow it down a bit!

Construction

The first requirements are the rectangular base board and two arms. Both arms must be the same lengths, and should be just slightly smaller than the width of the baseboard. The dimensions are a matter of little consequence, but about 15" (38cm) x 20" (51cm) is convenient for the base with arms 14" (35cm) long x 1" wide. The base may be plywood or chipboard, but the arms should be lightweight, thin plywood is suitable or some strips of plastic will do very well.

Also needed are 2 linear 10k potentiometers which should be good quality types. "Plastic" potentiometers are ideal but carbon pots give acceptable results although there may be distortions due to linearity errors. If possible, when you buy your pots, find out the "angle of electrical rotation" as this value will be required in the calculations.

Fig 1 shows how the arms need to be drilled to accept the potentiometers. Drill Arm 1 to accept a pot, shaft at one end, and a tight fit for a pot spindle at the other. Drill Arm 2 to a tight fit for a spindle at one end, and using Arm 1 as a template, mark the ▶

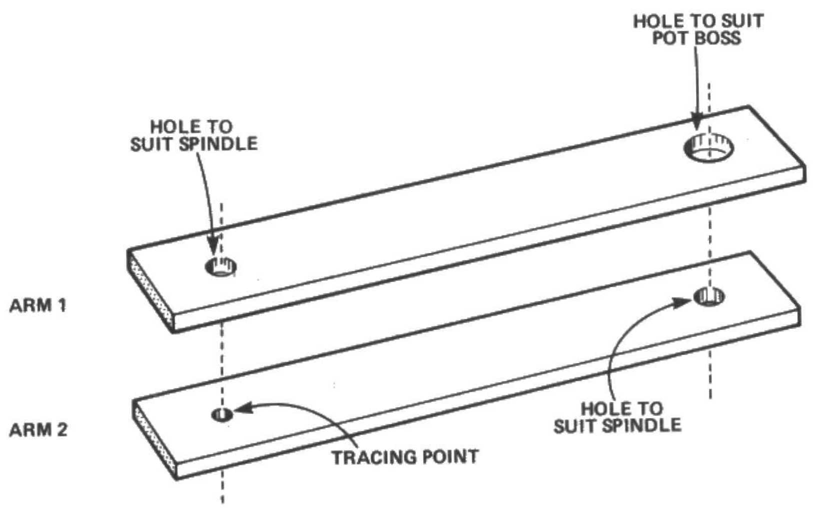


Figure 1. Detail of how the arms should be drilled to accept the potentiometers.

other end of the arm so that the effective length of both arms is identical. This mark is your "tracing point". If transparent plastic arms are used, a mark is all that is needed at the end of Arm 2, but if, for example, plywood is used, then cut a point at the end of the arm as shown in the diagram. Make sure that the underside of Arm 2 is very smooth, as it will have to slide over your diagrams.

The next task is to assemble the arms onto the baseboard.

First, stick one of the pots in the lower left hand corner of the baseboard (but about 1 inch from the edge), see **Fig 3.** Take care to orientate the connecting tags towards the corner as shown in the diagram. Use a good quality adhesive such as a "5-minute" epoxy. While the adhesive is setting, assemble the other potentiometer shaft into the hole in Arm 1 and tighten the locknut, making sure that the connecting tags point along the arm, then push Arm 2 onto the shaft. Remember, it needs to be a tight fit.

Now slide the arm onto the shaft of your baseboard ("fixed") potentiometer.

Refer to **Fig 2** and connect up the electrical circuits, by soldering directly onto the pots and solder buckets of the 15 ways plug. Twin screened cable can be used for each pot because it is freely available. But if you can find a convenient source of 6 way cable, that may be used instead. Note however, if unscreened cable is used, it is possible that some extra "noise" may be introduced to the system. Anchor your wires to the edge of the baseboard.

When wiring is complete, we are ready to calibrate the digitiser. To achieve this, we need to know the "electrical rotation" of the pots.

First, we must convert the rotation of the potentiometers in information the computer can accept. As we wish to do some trigonometry, the easiest form is to measure the rotation in radians. The potentiometers, give us a voltage, digitally converted via the ADVAL channels, to a number corresponding to their rotation which will be 65520 at maximum and zero at minimum. Thus if we know the angle of electrical rotation, we may ask the computer to calculate us a "magic number" which we may use in all future calculations. Type in:-

```
P. 65520 / (angle of rotation/DEG(1))
```

For 240 deg we get 15641.7478, which rounds up to 15642.

If you do not know the rotation of your pots, don't despair; the most common value is 270 deg, which gives us a magic number of 13904. Try that, it will probably work!

The routine that will do our calculation for the digitiser is:

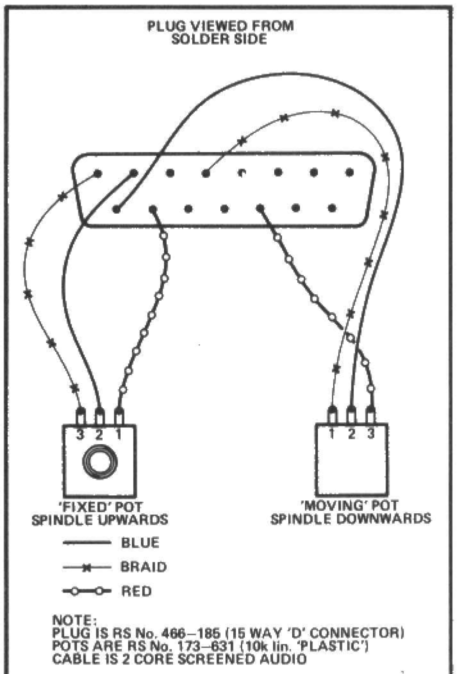

PLUG VIEWED FROM SOLDER SIDE
3 2 1
1 2 3
'FIXED' POT SPINDLE UPWARDS
'MOVING' POT SPINDLE DOWNWARDS
BLUE
BRAID
RED
NOTE:
PLUG IS RS No. 466–185 (15 WAY 'D' CONNECTOR)
POTS ARE RS No. 173–631 (10k lin. 'PLASTIC')
CABLE IS 2 CORE SCREENED AUDIO

Figure 2. Connections of the pots to the A/D port are made via three wires.

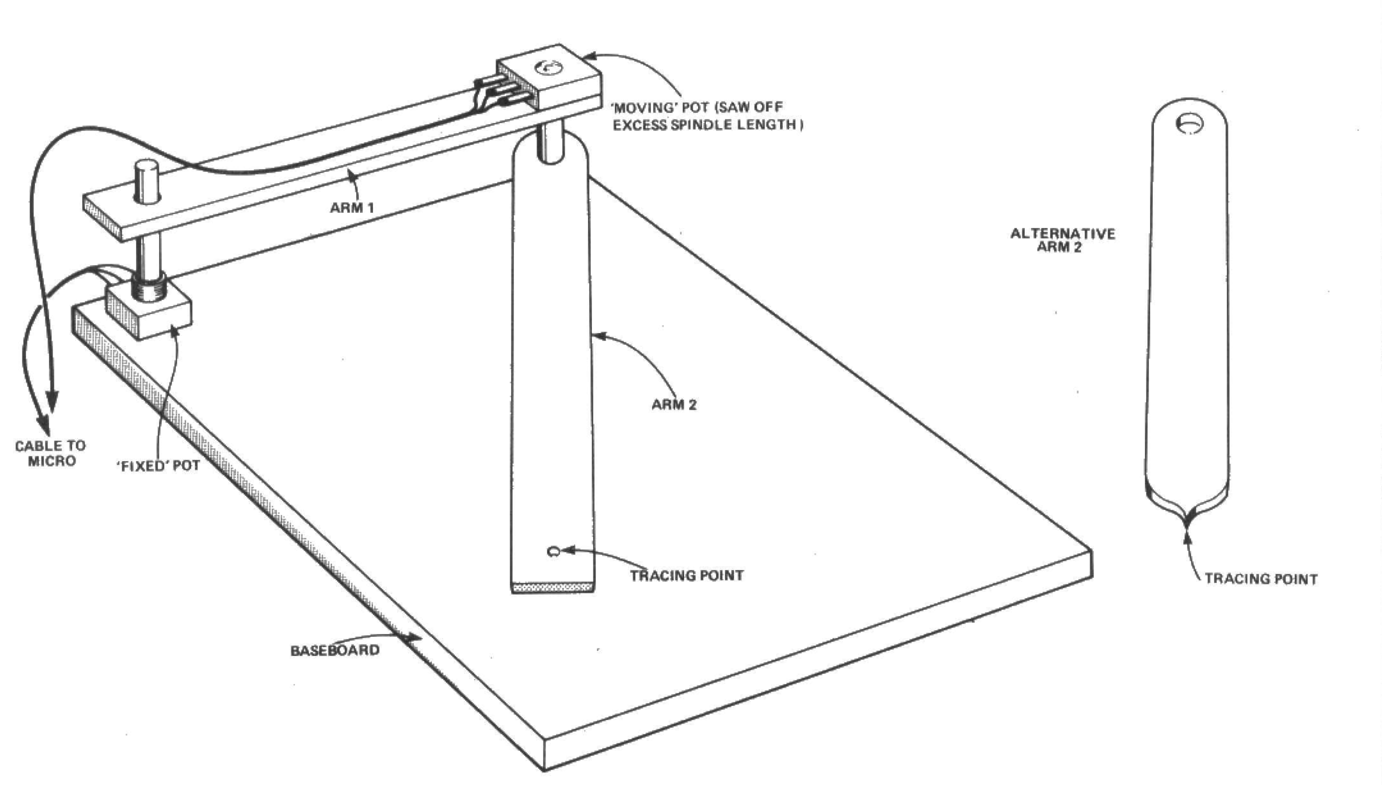


Figure 3. The general layout of the Digitiser showing the relationship between the various components of the graphics table.

```
10 THA = ADVAL(1)/15642:REM
   240 deg pot.
20 THB = ADVAL(2)/15642
30 THC=(THA+THB)
40 X=COS(THB):Y=SIN(THB)
50 XX=COS(THC):YY=SIN(THC)
60 XD%=INT(X-XX)*1000)
   -200:REM zero offset.
70 YD%=INT((Y-YY)*1000
```

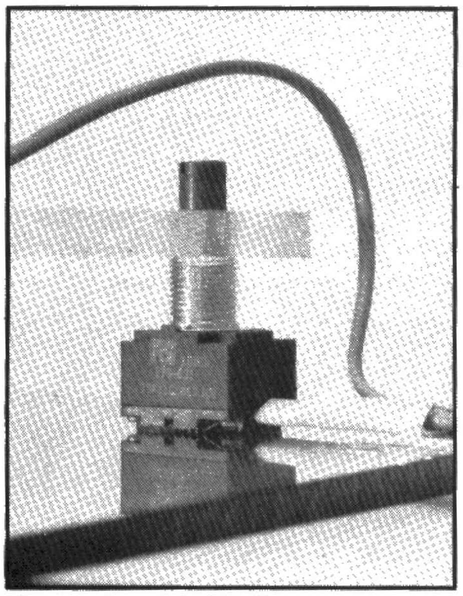
XD% and YD% are our screen co-ordinates corresponding to the "tracing points" position.

If you can't follow the trigonometry, don't worry, just use it.

To calculate our hardware, we need a fixed point on the baseboard, so as we know that the screen co-ordinates for centre of the screen are:

X=639, Y=511,

that is convenient to use. Now calculating back to find the corresponding ADVAL values is rather tedious, so conversion factors have been calculated to enable you to rapidly set up your arm. These are used in program 2, which is used in the set-up procedure.


Close up view of the 'fixed' pot.

Setting Up:

Before we can use the calibration program we must find the "centre point" of our display area. First fix a piece of paper to the baseboard, then, with a pencil, mark out the arc travelled by the potentiometer spindle at the end of Arm 1 as it is swept across the paper. Now hold Arm 1 parallel to the edge of the board, and sweep Arm 2 across the paper, marking the point where the "tracing point" cuts the arc. This is the "centre point" of your graphics area. It is worth indelibly marking this point upon your baseboard, in case you have to recalibrate in the future.

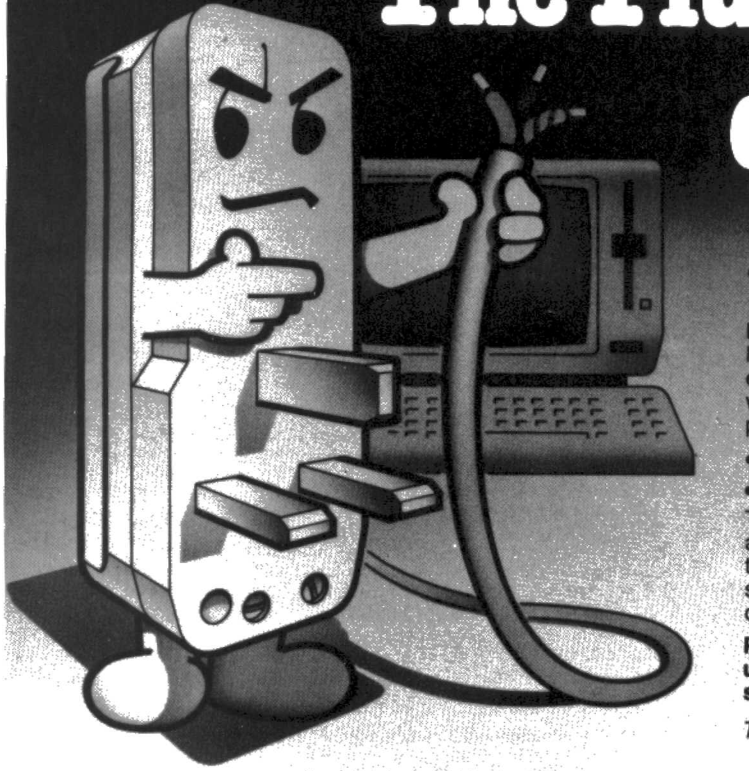
Now plug in, switch on, and type in and RUN program 1.

With the "tracing point" at the "centre point", adjust the potentiometer spindles without moving the arms until the numbers displayed on the screen correspond to the targets.

Do not worry about the last digit, the variations are due to the noise in the system, just do the best you can. Take care over this section, as the accuracy of your graphics depend upon it!

Now, 'lock' the spindles with a spot of adhesive, and you have only to type in program 2 (with the correct magic number) and your digitiser is ready to go! E&CM

The Plug says 'No' to dirty power.



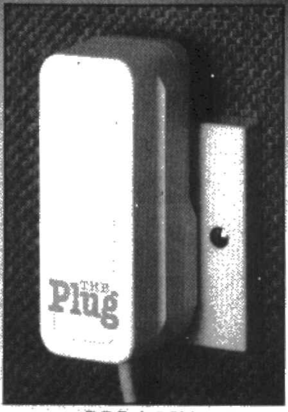
WHY

Because dirty power is the most likely cause of the unexplained errors and circuit malfunctions which can upset all micro based products and their users.

HOW

The Plug with its own built in filter and transient suppressor reduces the effect of electrical noise and spikes and increases the reliability of both hardware and software.

Protect your computer from unnecessary power problems, say 'No' to dirty power.



R.R.P. £15.50 including p.p.
Dealer enquiries welcome.

The Plug is available from dealers or direct from the manufacturer

WARNING: Unfiltered power can damage your computer's health.

FOR FURTHER INFORMATION CONTACT:
Power International Limited 2A Isambard Brunel Road, Portsmouth, Hampshire. PO1 2DU. Tel. (0705) 756715

FLEX EXPLAINED

Paul Izod and Alan Stirling look at the industry standard FLEX OS with particular reference to the *E&CM* hi-res computer system.

It has often been pointed out in this series of articles that FLEX only comes to life when disc drives are added to a system. Last month we described the disc controller card for the *E&CM* computer system and this month we look at the inplementation and operation of the FLEX OS on the system.

Before going into the details of FLEX, it is necessary however to discuss the general features of single-user operating systems. Since these all require the use of disk drives, a brief explanation of the means of data storage on a disk is appropriate.

The Data Format

The data areas on all disks are split up into circular tracks on the disk, with each of these tracks sub-divided into sectors – normally between 10 and 32 per track. Each sector holds between 128 and 1024 bytes of information. The smaller the number of bytes per sector, the more sectors per track. A 5¼" disk holds about 2,560 data bytes per track in single density. A 40 track single-sided/single-density 5¼" disk has a total capacity of 102,400 bytes. The start of each track is marked by an index hole in the disk near it's hub.

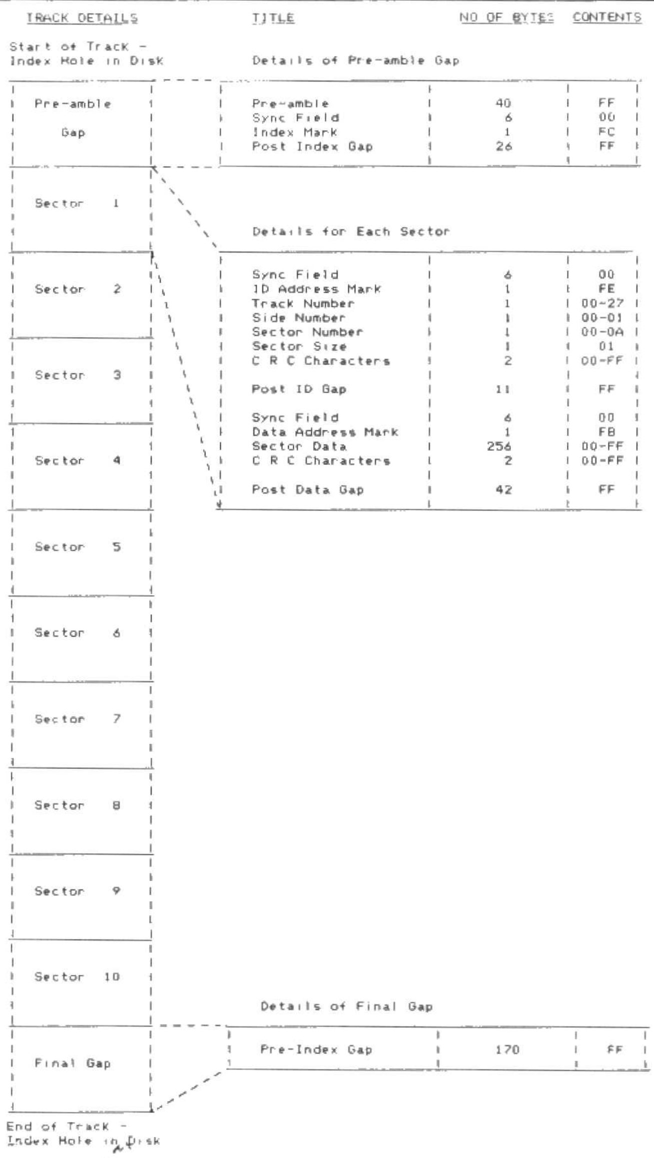
There are two systems used to correctly identify the sectors around each track. The most popular, called 'Soft Sectored', uses a small header area recorded onto the disk, in front of each sector, which holds the track and sector number of the sector that follows (see **Fig 1**). This information is recorded at the time that the disk is formatted, since without it the disk drive head assembly is not able to check its position on the disk. The other method, called 'Hard Sectored', used extra holes around the hub of the disk to mark the start of each sector. Extra circuitry counts pulses generated by these holes and generates the sector numbers. Soft sectored disks operate more reliably, since each sector is uniquely coded, but this is at the expense of total data capacity, as the header information takes up valuable data information space on the disk.

The interface between the computer system, and the disk drive is undertaken by a disk controller card. The key component on this board is the disk controller chip, which controls the operation of the disk drives completely. Since this chip is itself as complex as a small microprocessor, it only requires a few support chips to fully control most types of disk drives. It is the task of the disk controller card to communicate with the disk drive, reading or writing sectors of data as required. Only complete sectors of information are transferred.

Due to the intelligence of the disk controller chip, it is only necessary to write short machine code routines (called "Drivers") to gain access to any individual sector. With the addition of some form of disk drive selection hardware, it becomes a simple matter to read sectors from one disk drive and write them to another, thus copying data between two disks. Without a detailed record of where data is stored on each disk however, this basic system is unable to ensure that the sectors that it is reading contain the required data and the sectors that it is writing to do not already contain valuable information.

What is needed is a system of indexing the data stored on a particular disk. Although this index or directory can be held in memory, the obvious place to keep it is on the disk itself, alongside the data. In all disk operating systems, the start of this directory is defined, so that the machine knows where to start searching for the name allocated to the data for which it is searching. Alongside the name entry in the directory is further information about the data. This includes the track and sector where the data starts and finishes, the number of sectors that the data occupies and the type of data itself (i.e. machine code programs, text files, basic programs etc.). Data in this format is normally known as a file.

It is for the control and manipulation of these files and their



TITLE	NO OF BYTES	CONTENTS
Pre-amble	40	FF
Sync Field	6	00
Index Mark	1	FC
Post Index Gap	26	FF

TITLE	NO OF BYTES	CONTENTS
Sync Field	6	00
ID Address Mark	1	FE
Track Number	1	00-27
Side Number	1	00-01
Sector Number	1	00-0A
Sector Size	1	01
C R C Characters	2	00-FF
Post ID Gap	11	FF
Sync Field	6	00
Data Address Mark	1	FB
Sector Data	256	00-FF
C R C Characters	2	00-FF
Post Data Gap	42	FF

TITLE	NO OF BYTES	CONTENTS
Pre-Index Gap	170	FF

Figure 1. Standard IBM34 Format for 256 byte per sector for 5¼" floppy discs.

relevant directory entries that disk operating systems (DOS) have been developed. However since disks using the same DOS will normally be compatible, meaning that disks recorded on one systemn can be read by another – even if the two systems are produced by different manufacturers, most operating systems go further, providing a completely compatible software environment within each machine, so that programs written to run on one type of machine can be transferred on compatible disks to another machine, giving identical results when run.

Software Compatibility

This is obtained using standard routines within the DOS. The address of these routines and their functions is provided to the programmer within the documentation of the DOS. The addresses and functions of these routines do not change between different types of machines, providing the DOS is the same. Any differences between the machines hardware configurations, such as the address of the I/O port or memory mapped screen are taken into account when the DOS is implemented on that machine. During the implementation, machine code routines are used to link the DOS to the I/O routines of the machine. It is important that these routines are written such that they allow the DOS to perform in identical ways on differing machines. Programs written to link with the standard addresses defined in the DOS therefore should have the same results on any system. It is difficult however, to write sophisticated output routines without knowing the exact screen size and format. Thus the standardisation is limited to programs with simple I/O requirements – programs with complicated graphics requirements will only run on similar hardware, since this will be accessed directly by the program.

An Overview Of FLEX

FLEX was originally written in 1977 for systems using the Motorola MC6800 processor, by Technical Systems Consultants Inc, of Forest Hill, North Carolina, USA. In 1977 it was converted to run on the MC6809 and has become a standard for these machines. It is used on many makes of machine, including SWTPC, Positron, Gimix, Smoke Signal and Mororola, with implementations on Apple II, Tandy Colour Computer and Hitachi machines. With many users world-wide there is a wide range of system software with a number of good applications packages available. With the recent adaptation for the Dragon 32, this operating system will become even more popular.

FLEX Disk Formats

Although many other operating systems use sectors of 128 bytes, FLEX uses 256 bytes per sector. This gives 10 sectors per track on 5¼" single sided/single density (SS/SD) disks, or 15 sectors on 8" SS/SD. In FLEX the sectors are numbered from 01 (Hex) upwards and the first and outermost track is numbered 00 (Hex). **Table 1** gives details of the track, sector & capacities for the more popular disk formats.

The figures in this table reflect the total amount of data that can be stored in practice, since the theoretical maximum can never be reached since track zero on each disk is reserved for the directory and other information, sectors themselves only hold 252 bytes of data, and track zero is always recorded in single density, even on double density disks in order to maintain compatibility with single density systems.

FLEX File Format

Files can be of any length, and are made up from a collection of linked sectors. These are linked by the first two bytes of each sector, which hold the track and sector number of the next sector of the file. In this way a file can occupy any unused sectors on the disk, since successive sectors do not have to be physically consecutive. The last sector of a file has these two link bytes set to zero. On most files the following two bytes hold the consecutive sector number within the file, starting with sector 1 as the first sector. Since each sector has 256 bytes and four are used by the system, there remain 252 bytes available to the user. This slight loss in storage capacity is outweighed by the flexibility of file layout on the disk. Even the directory information on track 0 uses the same format.

All empty or unused sectors on the disk are linked as one large file, called the 'Free Chain'. This file also uses the track and sector links, but does not maintain a consecutive sector number.

When a file is to be written to the disk, FLEX removes the first sector from the free chain, and allocates it as the first sector of the new file. As the new file requires more sectors, FLEX re-allocates them from the free chain to the new file. They are already linked toghether, since they were linked in the free chain. At the end of the file, FLEX un-links the last sector of the file, (track and sector equal to zero), updates the directory entry for the file, and and changes the pointer to the start of the free chain. If a file is deleted, it is linked to the end of the free chain and it's entry is deleted from the directory. Although it has been deleted, with it's directory entry removed, the data will still be intact, until overwritten by another file. The disk is full when the free chain has no further free sectors.

TABLE 1 – Track, Sector and Byte Capacities of Different Sizes of FLEX Disks.

Disk Size	No. of Sides	Data Density	No. of Tracks	Largest Track Track	Largest Sector (Dec/Hex)	Total Disk Secotrs	Total Data Sectors	Total Data Bytes
5"	1	Single	40	39/27	10/0A	400	390	98280
5"	1	Double	40	39/27	18/12	712	702	176904
5"	2	Single	40	39/27	20/14	800	780	196560
5"	2	Double	40	39/27	36/24	1424	1404	353808
5"	1	Single	80	79/4F	10/0A	800	790	199080
5"	1	Double	80	79/4F	18/12	1432	1422	358344
5"	2	Single	80	79/4F	20/14	1600	1580	398160
5"	2	Double	80	79/4F	36/24	2864	2844	716688
8"	1	Single	77	76/4C	15/0F	1155	1140	287280
8"	1	Double	77	76/4C	26/1A	1991	1976	497952
8"	2	Single	77	76/4C	30/1E	2310	2280	574560
8"	2	Double	77	76/4C	52/34	3982	3952	995904

(Dec: Decimal, Hex: Hexadecimal)

Track Zero Information

This track has a special significance since it holds the disk directory and certain other important information.

The first sector holds the boot loader program. This program is machine specific, and is called into the machine by the monitor program, when FLEX is booted. Once loaded into the machine, it loads FLEX into memory, from wherever it is located on the disk. Finally, it passes control to FLEX, which begins operation. If the boot program is too large to fit into one sector, the second sector may also be used.

The third sector is called the 'System Information Record' (SIR). This holds information regarding the disk itself, and the chain of free sectors on the disk. See **Table 2.** This information is read by FLEX prior to accessing a disk file, so that FLEX can adapt to the individual disk's format.

TABLE 2 – List of Information held on System Information Record (SIR).

Volume Name of Disk	(8 Characters)
Volume Number of Disk	(1 – 65535)
Creation Date	(MM/DD/YY)
Start of Free Chain	(Track/Sector)
End of Free Chain	(Track/Sector)
Length of Free Chain	(No of Sectors)
Highest Sector Address on Disk	(Track/Sector)

Sector 4 is left blank, and is reserved for future expansion.

The directory file starts at sector 5, and continues through all the other sectors on track zero. Each directory entry takes 21 bytes, giving 12 entries per sector. If more directory entries are required, over and above those available on track zero, FLEX automatically allocates extra sectors as required from the free chain.

FLEX File Specifications

As with all operating systems, there is a structured syntax to be complied with, when entering the details of a disk file to be used. FLEX numbers the disk drives from '0' to '3', this number being used at the beginning of the file name. The file name itself must not be more than 8 characters. FLEX also allows files to use extension names of up to three letters, in order to describe the file type. The main extensions used are:

BIN	Binary Program Files
TXT	Text files
CMD	FLEX Command Files
BAS	Basic Source Code Files
SYS	FLEX System Files
BAK	Back-up Copy of Text File
BAC	Basic Compiled (Tokenised) Files
OUT	Printer-Spooling Output Files

The drive number normally precedes the file name, whereas the extension follows it. They are separated by a full stop, thus '1.LETTER.TXT', is a text file on drive '1', called 'LETTER'. Many commands presume default options in terms of drive number and extension, thus in most situations it is only necessary to enter the file name alone.

FLEX Command Structure

There are only two memory resident commands within FLEX, although others can be added if required. 'GET' is used to load a binary file into memory, where it will be loaded into the same memory locations from which it was saved. 'MON' is used to exit from FLEX, back to the system monitor. All other commands are loaded from disk. To initiate a command, simply enter the name of a command file. FLEX will look for a file with the same name, on the system disk, (normally Drive '0'), and with the extension of '.CMD', signifying a command file. Thus entering 'CAT' will loade the catalogue command from drive '0'. Its full file specification is '0.CAT.CMD'. Any of the default options can be changed merely by entering the option desired. For example to run a binary program on drive '1', called 'TEST', enter '1.TEST.BIN'.

The system also controls the default options on two drives, known as the 'System Drive' and the 'Working Drive'. The system drive is where FLEX looks for command files, and the working drive for text and other files. Thus if the system drive is set to '0' and the working drive to '1', the command 'LIST LETTER', will load the command file '0.LIST.CMD', which will in turn list on the screen the contents of the text file '1.LETTER.TXT'. These default drive numbers can be controlled using the 'ASN' (Assign) command. Using this same philosophy, many commands use a string of parameters entered after the command name in order to control the detailed operation of the command.

Within this simple framework, it is possible to control the operation of a sophisticated 6809 based computer, particularly if the range of commands is augmented, which is done by just adding the command file to the system disk. E&CM

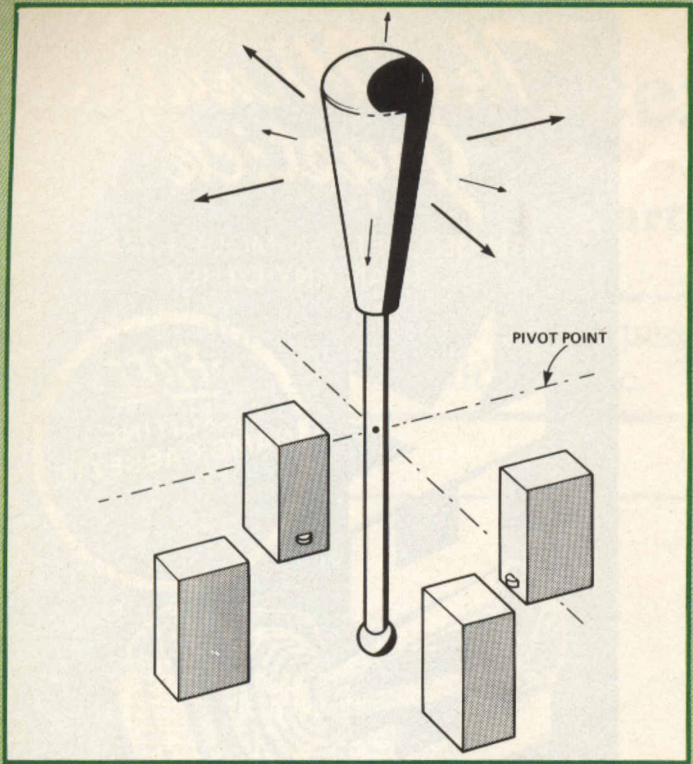
Next Month: More on Flex and the *E&CM* Hi-Res Computer.

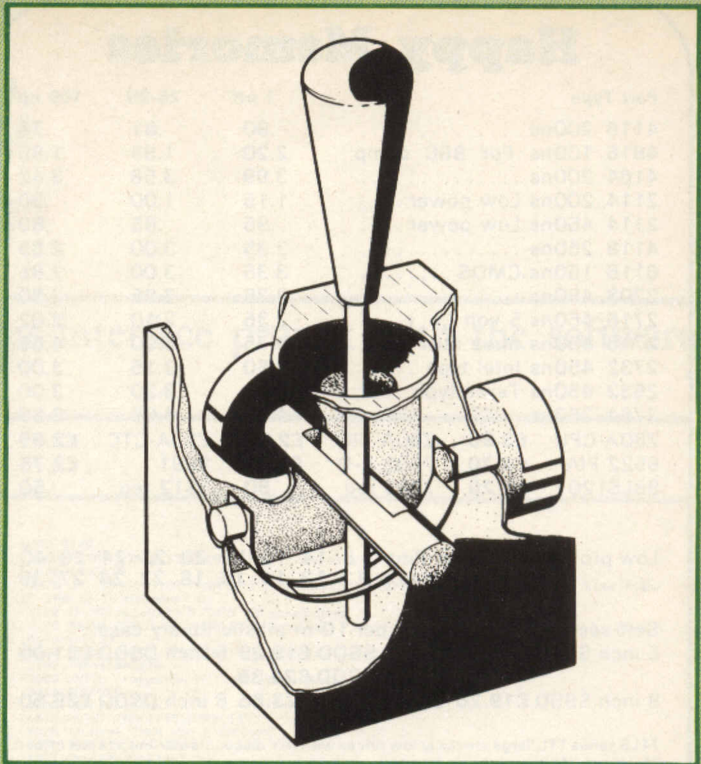
BBC Sequencer Interface Part 2

Robert Penfold concludes his sequencer interface project with the software associated with last month's article.

```
   10 REM SEQUENCER PROGRAM
   20 REM by John Penfold
   30 REM Copyright 1983
   40 ON ERROR GOTO 300
   45 ?&FE62=127:?&FE6C=160
   50 DIM M%(1000,2):REM music storage array
   60 N%=0:REM note counter
   70 CLS:VDU 23,1,0;0;0;0;
   80 PRINT TAB(5,5)CHR$(132);CHR$(141);"Do you wish to :-"
   90 PRINT TAB(5,6)CHR$(132);CHR$(141);"Do you wish to :-"
  100 PRINT TAB(1,9)CHR$(131);"1. Enter music from the computer"
  110 PRINT TAB(1,11)CHR$(131);"2. Load music from tape"
  120 PRINT TAB(1,13)CHR$(131);"3. Review/alter a note or time"
  130 PRINT TAB(1,15)CHR$(131);"4. Play music once"
  140 PRINT TAB(1,17)CHR$(131);"5. Play music as a loop"
  150 PRINT TAB(1,19)CHR$(131);"6. Record music on tape"
  160 PRINT TAB(5,21)CHR$(132);"Enter number only"
  170 choice$=GET$
  180 choice=VAL(choice$)
  190 IF (choice <1) OR (choice >6) THEN VDU 7:GOTO 170

  200 IF choice> 6 THEN VDU 7:GOTO 170
  210 IF choice=1 PROCenter
  220 IF choice=2 PROCload
  230 IF choice=3 PROCchange
  240 IF choice=4 PROCplay
  250 IF choice=5 PROCloop
  260 IF choice=6 PROCsave
  270 GOTO 70
  280 END
  300 IF ERR=14 THEN CLS:PRINT TAB(5,10)CHR$(129);"MEMORY NOW FULL":PRINT TAB(1,
12)CHR$(131);"Press any key to continue":REPEAT UNTIL GET:N%=N%-1:GOTO 70
  310 IF ERR=15 CLS:PRINT TAB(5,10)CHR$(129);"MEMORY NOW FULL":PRINT TAB(1,12)CH
R$(131);"Press any key to continue":REPEAT UNTIL GET:GOTO 70
  320 IF (ERL>=6180) AND (ERL<=6230) THEN CLOSE#M:GOTO 70
  330 IF (ERL>=2080) AND (ERL<=2130) THENCLOSE#M:GOTO 70
  340 IF ERR=17 AND (ERL>=1090) AND (ERL<=1180) THEN N%=N%-1:GOTO 70
  350 IF ERR=17 GOTO 70 ELSE CLS:REPORT
  360 END
 1000 DEF PROCenter
 1010 CLS
 1020 PRINT TAB(5,8)CHR$(131);"Do you wish to add to old music"
 1030 PRINT TAB(5,11)CHR$(131);"or start a new piece?(O or N)"
 1040 clear$=GET$
 1050 IF (clear$="N") OR (clear$="n") THEN N%=0
 1070 REPEAT:REM note entry start
 1080 CLS
 1090 N%=N%+1
 1100 PRINT TAB(3,5)CHR$(132);"Note no. ";N%
 1110 PRINT TAB(3,7)CHR$(132);"Enter the pitch value"
 1120 INPUT TAB(3,8),pitch
 1130 IF(pitch<1) OR (pitch>63) THEN VDU 7:PRINT TAB(4,8)" ":GOTO 1120
 1140 M%(N%,1)=pitch
 1150 PRINT TAB(3,9)CHR$(132),"Enter the time value"
 1160 INPUT TAB(3,10),time
 1170 IF time<5 THEN time=5
 1180 M%(N%,2)=time
 1190 PRINT TAB(1,15)CHR$(131);"Note no.  ";N%;": Pitch ";M%(N%,1);": Time ";M%(
N%,2)
 1200 PRINT TAB(1,17)CHR$(131);"Another note? (Y or N)"
 1210 repeat$=GET$
 1220 UNTIL (repeat$="N") OR (repeat$="n")
 1230 ENDPROC
 2000 DEF PROCload
 2010 CLS
 2020 PRINT TAB(5,8)CHR$(131);"Do you wish to add to old music"
 2030 PRINT TAB(5,10)CHR$(131);"or load a new piece? (O or N)"
 2040 clear$=GET$
 2050 IF (clear$="N") OR (clear$="n") THEN N%=0
 2060 PRINT TAB(5,13)CHR$(131);"Enter filename of music "
 2070 INPUT name$
 2080 M=OPENIN name$
 2090 REPEAT
 2100 N%=N%+1
 2110 INPUT# M,M%(N%,1),M%(N%,2)
 2120 UNTIL EOF# M
 2130 CLOSE#M
 2140 ENDPROC
 3000 DEF PROCchange
 3010 CLS:IF N%=0 VDU 7:ENDPROC
 3020 PRINT TAB(5,8)CHR$(131);"Selective review of a note(S)"
 3030 PRINT TAB(5,10)CHR$(131);"Or global alteration of tempo(T)"
 3035 PRINT TAB(5,12)CHR$(131);"Or Edit (insert/delete) a note?(E)"
 3040 choose$=GET$
 3050 IF (choose$="T") OR (choose$="t") THEN PROCtempo:ENDPROC
 3055 IF (choose$="E") OR (choose$="e") THEN PROCedit:ENDPROC
 3060 REPEAT
 3070 CLS
 3080 PRINT TAB(3,5)CHR$(131);"Which note do you wish to review?"
 3090 INPUT TAB(3,6),note%
 3095 IF note%<1 THEN note%=1
 3100 IF note%>N% THEN PRINT TAB(3,7)CHR$(132);"Highest note no. is ";N%:PRINT T
AB(3,9)CHR$(131);"Press any key to continue":a$=GET$:GOTO 3070
 3110 PRINT TAB(3.7)CHR$(132);"Note no. ";note%;": Pitch ";M%(note%.1);": Time "
;M%(note%,2)
 3120 PRINT TAB(3.9)CHR$(131)"Alter pitch? (Y or N)"
 3130 p$=GET$
 3140 IF (p$="Y") OR (p$="y") THEN PRINT TAB(3,11)CHR$(131)."Enter new pitch ":I
NPUT TAB(21,11)M%(note%,1)
 3145 IF (M%(note%,1)<1) OR (M%(note%,1)>63) THEN VDU 7:PRINT TAB(21,11)"     ":
GOTO 3140
 3150 PRINT TAB(3,12)CHR$(131);"Alter time? (Y or N)"
 3160 t$=GET$
 3170 IF (t$="Y") OR (t$="y") THEN PRINT TAB(3,13)CHR$(131)"Enter new time ":INP
UT TAB(19,13)M%(note%,2)
 3172 IF M%(note%,2)<5 THEN M%(note%,2)=5
 3175 PRINT TAB(3,15)CHR$(132)"Note no. ";note%;": Pitch ";M%(note%,1);": Time "
;M%(note%,2)
 3180 PRINT TAB(1,17)CHR$(131);"Another note? (Y or N)"
 3190 repeat$=GET$
 3200 UNTIL (repeat$="N") OR (repeat$="n")
 3210 ENDPROC
 4000 DEF PROCplay
 4010 PRINT TAB(1,15)CHR$(133)
 4020 IF N%=0 THEN VDU 7:ENDPROC
 4030 FOR play%=1 TO N%
 4035 ?&FE60=M%(play%,1)
 4040 T=TIME
 4050 REPEAT UNTIL TIME=T+M%(play%,2)
 4060 NEXT play%
 4070 ENDPROC
 5000 DEF PROCloop
 5010 PRINT TAB(1,17)CHR$(133)
 5020 PRINT TAB(5,21)CHR$(133);"Press any key to stop"
 5025 PRINT TAB(5,22)CHR$(133);"(at end of loop)"
 5030 IF N%=0 THEN VDU 7:ENDPROC
 5040 REPEAT
 5050 FOR play%=1 TO N%
 5060 ?&FE60=M%(play%,1)
 5070 T=TIME
 5080 REPEAT UNTIL TIME=T+M%(play%,2)
 5090 NEXT play%
 5100 UNTIL INKEY$(1)<>""
 5110 ENDPROC
 6000 DEF PROCsave
 6010 CLS:IF N%=0 THEN VDU 7:ENDPROC
 6020 PRINT TAB(5,8)CHR$(131);"Save all music in memory"
 6030 PRINT TAB(5,10)CHR$(131);"or a selected part? (A or S)"
 6040 choose$=GET$
 6050 IF (choose$<>"S") AND (choose$<>"s") THEN start%=1:end%=N%:GOTO 6150
 6060 PRINT TAB(5,13)CHR$(132);"Last note is no. ";N%
 6070 PRINT TAB(1,15)CHR$(131);"Enter first note you wish to save"
 6080 INPUT TAB(1,16),start%
 6090 IF start%<1 THEN start%=1
 6100 IF start%>N% THEN VDU 7:PRINT TAB(1,16)"     ":GOTO 6080
 6110 PRINT TAB(1,17)CHR$(131);"Enter last note you wish to save"
 6120 INPUT TAB(1,18)end%
 6130 IF end%>N% THEN end%=N%
 6140 IF end%<start% THEN VDU 7:ENDPROC
 6150 PRINT TAB(1,20)CHR$(131);"Enter the filename of the music"
 6160 PRINT TAB(1,22)CHR$(131);"Up to 10 letters, no spaces."
 6170 INPUT name$
 6180 M=OPENOUT name$
 6190 FOR notes=start% TO end%
 6200 PRINT# M,M%(notes,1),M%(notes,2)
 6210 NEXT notes
 6230 CLOSE# M
 6240 ENDPROC
 7000 DEF PROCtempo
 7010 CLS
 7020 PRINT TAB(3,5)CHR$(131);"Do you wish to make the"
 7030 PRINT TAB(3,7)CHR$(131);"Tempo faster or slower (F or S)"
 7040 select$=GET$
 7050 IF (select$<>"F") AND (select$<>"f") AND (select$<>"S") AND (select$<>"s")
 THEN VDU 7:GOTO 7040
 7060 PRINT TAB(3,11)CHR$(131);"Enter the factor by which"
 7080 PRINT TAB(3,13)CHR$(131);"you wish to alter the tempo"
 7090 INPUT TAB(3,14),factor
 7095 IF factor<=1 THEN PRINT TAB(3,14)"     ":VDU 7:GOTO 7090
 7100 IF (select$="F") OR (select$="f") THEN PRINT TAB(3,17)CHR$(132);"FASTER by
 a factor of ";factor
 7110 IF (select$="S") OR (select$="s") THEN PRINT TAB(3,17)CHR$(132);"SLOWER by
 a factor of ";factor
 7120 FOR notes=0 TO N%
 7130 IF(select$="F") OR (select$="f") THEN M%(notes,2)=M%(notes,2)/factor:IF M%
(notes,2)<5 THEN M%(notes,2)=5
 7140 IF(select$="S") OR (select$="s") THEN M%(notes.2)=M%(notes,2)*factor
 7150 NEXT notes
 7160 ENDPROC
 8000 DEF PROCedit
 8010 CLS
 8020 PRINT TAB(5,8)CHR$(131);"Do you wish to"
 8030 PRINT TAB(5,10)CHR$(131);"Insert or Delete? (I or D)"
 8040 id$=GET$
 8050 PRINT TAB(1,12)CHR$(131);"Enter the position at which"
 8060 PRINT TAB(1,14)CHR$(131);"you wish to insert/delete"
 8070 INPUT TAB(1,15)note%
 8080 IF note%<1 THEN note%=1
 8090 IF note%>N% THEN PRINT TAB(3,17)CHR$(132)"Highest note no. is";N%:PRINT TA
B(3,19)CHR$(131)"Press any key to continue":A$=GET$:GOTO 8010
 8100 IF (id$="I") OR (id$="i") THEN FORinsert=N% TO note% STEP-1:M%(insert+1,1)
=M%(insert,1):M%(insert+1,2)=M%(insert,2):NEXT:N%=N%+1
 8110 IF (id$="D") OR (id$="d") THEN FORdelete=note% TO N%:M%(delete,1)=M%(delet
e+1,1):M%(delete,2)=M%(delete+1,2):NEXT:N%=N%-1:ENDPROC
 8120 CLS:PRINT TAB(1,5)CHR$(131);"Enter new pitch"
 8130 INPUT TAB(1,6),M%(note%,1)
 8140 IF (M%(note%,1)<1) OR (M%(note%,1)>63) THEN VDU 7:GOTO 8120
 8150 PRINT TAB(1,8)CHR$(131);"Enter new time"
 8160 INPUT TAB(1,9),M%(note%,2)
 8170 IF (M%(note%,2)<5) THEN M%(note%,2)=5
 8180 ENDPROC
```

E&CM





JOYSTICKS

Ken Alexander looks at some examples of the most popular computer peripherals.

While nobody would argue that the majority of joysticks are bought for use with games of one sort or another, the humble joystick can have more serious uses. While not many of you will want to run CAD/CAM packages on your micro, the area of computer control of mechanical systems is something that many more people are becoming involved in. In this sort of application, inputting commands via a joystick control can prove for more effective than keyboard entry. It would be a mistake, therefore, to dismiss the joystick as something for kids, grown up or otherwise.

With the above thoughts in mind, we decided it was time for *E&CM* to take a look at some of the more popular joysticks.

Port in a Computer

The computer hardware industry is not known for any attempt at standardisation and it is nor surprise that joysticks come in a variety of incompatible configurations. The nearest thing to a standard in this non-standard area is the Atari/Vic format for switch type joysticks. The machines from these two manufacturers were, because of their games orientated design, among the first to incorporate a joystick port as part of the computer's basic specification. Other manufacturers are now starting to realise that the addition of a joystick port can add to the overall attractiveness of their products. The Dragon 32 and Sord M5 are examples of recent designs that feature dedicated joystick ports while the Comex goes one stage further and incorporates a joystick within the machines keyboard. None of these computers adopt the Atari/Vic format and the diversity of designs seems bound to increase in the future.

Interfaces

Of those computers that do not provide a dedicated joystick port many feature a user port which can be used to input information from a joystick – perhpas the best example of this is the BBC Model B with its A/D convertors.

Other machines, without even a user port, demand that a joystick is to be used. Interface designs adopt two different approaches to the problem of connecting the joystick to a computer. Either the interface mimics the action of the keyboard – thus allowing joysticks to be used with software written for keyboard use without any modification or, the interface pokes joystick status information into a special location in the computer's memory which must then be read by a peek from within the software.

Analogue and Digital

Joystick designs can be split into two distinct types. The first of these is the switch type of stick. In this sort of design, the joystick consists of a pivoted stick the lower end of which is surrounded by four microswitches. Moving the stick to one of the four major directions will close one of the switches and place a unique bit pattern onto the computer joystick port. In some designs, if the stick is moved to an intermediate direction, two switches are closed and suitable software can

Illustration Top Left:

An exploded view of a switch type joystick typified by Atari sticks. The stick is pivoted centrally and is surrounded by four microswitches at its lower end. Moving the stick to and off the four major points will cause one or other of the switches to close. In some designs, moving the stick to an intermediate point will close two switches to provide another four decodeable positions.

This type of joystick can only offer a limited resolution.

Illustration Top Right:

View of a potentiometer style joystick. Two potentiometers are mounted on opposing axes with each providing a voltage output proportional to its position on one of the axes. The two outputs when decoded can uniquely identify any point on the X-Y plane.

The accuracy of this joystick is limited by the potential applied to the track of the potentiometers and by the accuracy of the A/D converter used to decode their position.

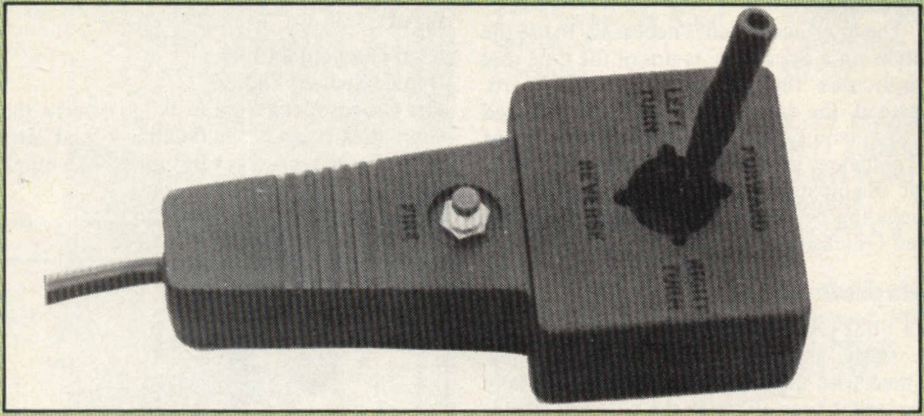
decode the new pattern produced and take appropriate action. It can be seen that this type of design offers a very coarse resolution with only four, or perhaps eight, decodable inputs.

The other major type of joystick is based on two potentiometers. By applying a known voltage across a potentiometer the value of the voltage appearing at the potentiometer's wider contact will be directly proportional to the position of the shaft. A single potentiometer can give an indication of position on a single axis while two potentiometers mounted at right angles to each other can give two outputs which, if decoded, can uniquely define any point in the X-Y plane.

Its apparent that in order to make use of a potentiometer type joystick, it must be used in conjunction with an A/D converter. Its also apparent that the accuracy of joystick based on potentiometer, depends on the stability of the voltage applied across the potentiometer and on the accuracy of the A/D unit.

Self Centred

Another major aspect of joystick design is whether or not the stick is self centering. Self centering designs usually offer a far more positive operation when playing games although in other applications the self centering action can be a disadvantage. Of those joysticks we tested, those with a self centering action were more robust than most and some would not be out of place in a professional arcade game.



Voltmace

Dragon 32 (Analogue) £5.00
Atari (Switch) £5.95 (£4.95 Kit)
BBC £10.95 (Adaptor £13.95)
The **Dragon 32** joystick from Voltmace is an analogue joystick with separate fire button. The potentiometer assembly is housed in a black plastic case shaped rather like the old fashioned carbon microphones. The moulding for this case seems to have done the rounds of many joysticks manufacturers as the Cascom/Clares/Microdeal/Midwich sticks are all housed in very similar housing.

The action of the stick was light and smooth and, using a two handed grip, accurate positioning of an on screen image and easy use of the fire button was possible. The stick is supplied with about 2.5 feet of cable and is terminated in the standard 5 pin DIN Dragon plug.

The **Atari** stick is housed in a case that is the same shape and size as the Dragon stick, the only variation on the theme being a set of embossed legends annotating the various control functions of the stick. Despite appearances, the Atari stick is a switch type although the feel is very much that of an analogue type. There is no positive 'switch' action but this should not detract from the stick's performance.

The joystick is supplied with a slightly longer lead than the Dragon sample and is terminated in the Atari standard 9 way D plug.

This joystick is availabe in kit form and while assembly would require basic soldering skills, the kit is very straightforward to put together.

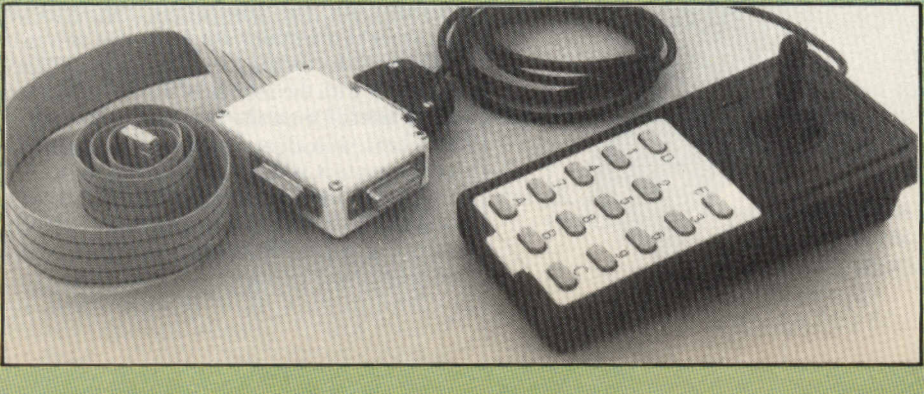
The **BBC** joystick plus keypad is a very impressive piece of hardware. Designated the D14/B, the stick is an analogue design that, in addition to the controller stick, offers 14 keys. The controller can either be plugged directly into the BBC model B or can be used with the D14/B/I adaptor if its full potential is to be realised.

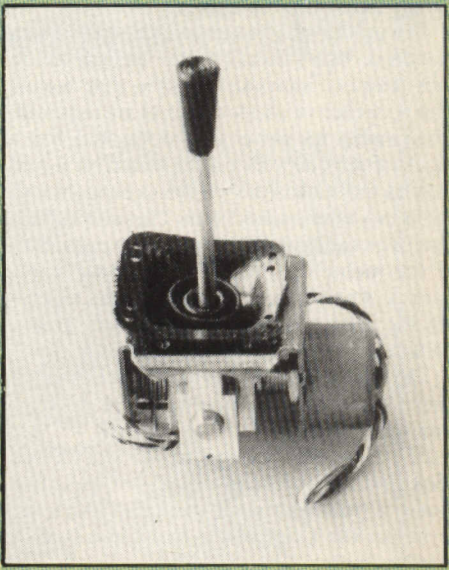
The D14 is housed in a black, calculator style box and the stick is of the self centering type. The potentiometers used in the assembly are high quality types with graphite wipers, fed with a 5 volt supply which together with a shunt resistor gives close to 0.9 volts in the centre position while using the full A/D converter range. This makes the sticks compatible with existing software.

The joystick, or perhaps more accurately, handset, can be plugged directly into the BBC's 15 way A/D port in which case ADVAL(1) and ADVAL(2) will return the position of the X and Y potentiometers. The buttons can be interrogated by the command ADVAL 0 AND 3=X. For no button pressed X will be 0, for one top row button X=1, for a second row button X=2 and any top and second row button pressed together will yield a value of 3 for X.

The handset comes into its own when used with the adaptor box which plugs into the A/D socket and into the user port. This port strobes the button columns with PB 4-6 and reads back the four rows on PB 0-30 the remaining line, PB 7 is used to select to a read of either the Left or Right handset. This joystick offers tremendous potential for the more adventurous programmer yet, with a couple of *FX commands, the controller maintains compatibility with software written for standard joysticks.

This handset is a cut above the crowd, both in the action of the self centering stick itself and in the flexibility of the keypad.

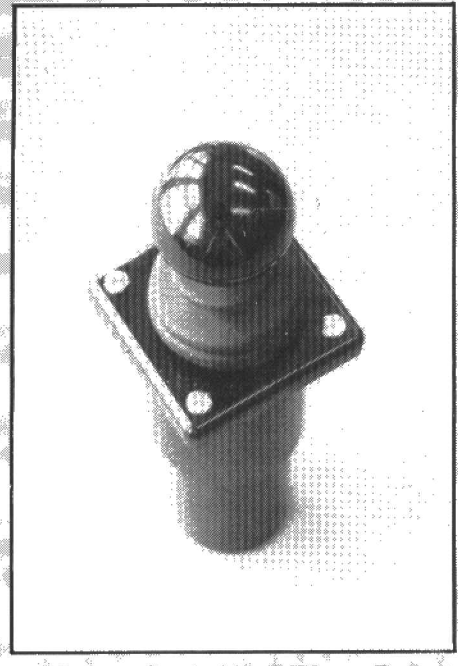




Flight Link Control

Dragon/Atari/BBC (Pair) £8.95 each
Dual axis potentiometer £2.00
Inductive dual axis unit £25.00

The cased joystick from Flight bore a marked resemblance to the general group of sticks mentioned above. The version ▶



supplied was fitted with a DIN type **Dragon** plug although the company supply similar units for both **Atari** type systems and a pair of units suitable for connection to the BBC Model B's A/D port.

The only difference between this unit and the Voltmace joystick was that the stick itself was rather shorter and the feel of the movement slightly stiffer.

Flight also supplied two 'bare' units. The first of these was a high quality, dual control axis potentiometer assembly that featured a self centering action. The standard of construction evident was extremely high.

The second stick was of the contactless inductive sensing type. This device is a professional controller that boasts a life in excess of 10 million cycles.

The unit 'looks' to the outside world like a standard, two potentiometer design but in fact features internal circuitry that means that no switch or wiper contacts are required. Presumably the lower end of the stick has a block of ferrous material mounted on it and that, as the stick is moved this is brought into proximity with a set of coils. These coils form part of an oscillator circuit and, the proximity of the metal will alter the oscillator's frequency of oscillation. Smoothing and rectifying the output of the oscillator will give a voltage proportional to the sticks position.

The feel of the unit was about the best of all sticks tested and the quality of the product was very evident.

Flight produce a range of versions of this stick and of more conventional types. For those of you who want to make your own joysticks the Company's catalogue is well worth a look.

Clares

Dragon 32 (Analogue £15.95 (Pair)
BBC (Analogue) £17.95 (Pair)
These joysticks are the familiar analogue design with the sticks for the BBC being wired to the 15 pin A/D connector.

Cascom

Dragon 32 (Analogue) £14.95 (Pair)
Very similar in all respects to the Clares/Voltmace etc. joysticks.

Kempston

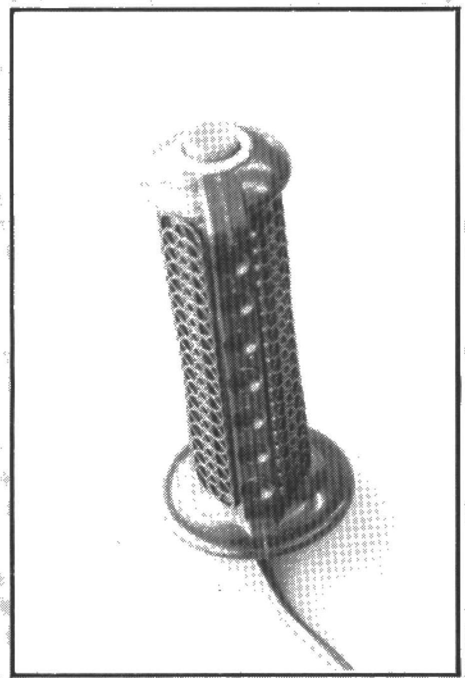
Spectrum (Switch)
£25.00 (including interface)
One of the best selling joysticks that is on the switch type (8 positions decoded) and two firing buttons. The joystick was very responsive and was of a very robust and attractive appearance.

The interface which is necessary to use the sticks on a Spectrum, is not of the type that duplicates the action of the keyboard. Instead, the status of the joystick is poked into a memory location. Fortunately, a range of software, some 14 in all has been written for Kempston joystick compatability – including best sellers from Artic, Silversoft and Quicksilva.

Mapsoft

Le Stick (Atari) £24.95
A fairly unusual and expensive joystick. Fitted with the Atari plug, the joystick is of the switch type although the switches are not the usual microswitches. Instead 'tilt' switches are used and mean that the stick is used by tilting the whole stick in the direction required. Squeezing the stick will halt its action.

Le Stick takes a little getting used to but, once acclimatised with the action, it is a very responsive and fun to use device. One minor point that we felt might warrant attention was that the stick could not be put down on a surface in its upright, inactive position, as the wire emerging from the base causes it to topple over. This was a handicap when using Le Stick with, for example, AGF's interface. The problem was that, as the joystick was active, it was locking out the computer's keyboard.



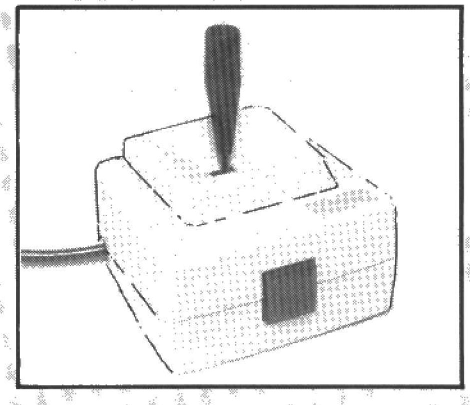
Cambridge Computing

Spectrum/ZX81 (Switch)
£26.00 (including interface)
Jupiter Ace (Switch)
£26.00 (including interface)
A switch type stick in the obligatory 'BS' case. Eight positions are decoded and, a la Kempston, two fire buttons are provided.

The Cambridge interface, duplicates the action of the keyboard of the host machine and can therefore be configured to work with any software package.

Kraft

Atari (Switch) £13.95
Apple (Switch) £42.00
The two joysticks were of a high quality, the Atari stick being of the familiar format. The stick offered a good performance, with quick and sensitive control.



Midasonic

Dragon (Analogue) £17.50 (Pair)
BBC (Analogue) £17.50 (Pair)
Good quality and attractively cased joysticks of the potentiometer type. These joysticks are designed to be used on a surface rather than being handheld.

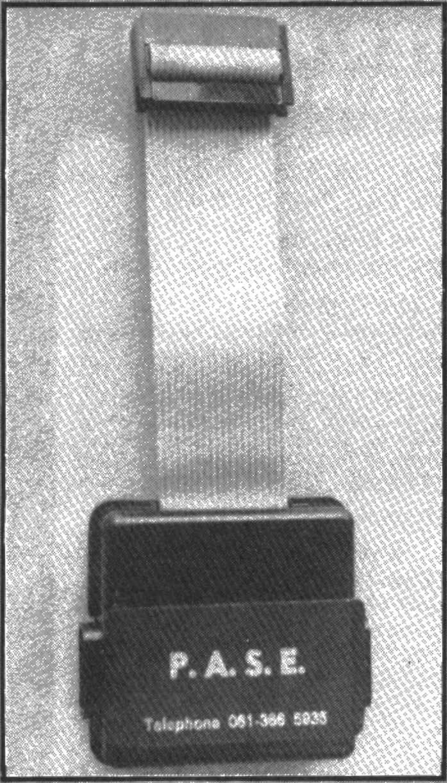
Midasonic are shortly to produce a joystick for use with Spectrum.

AGF Hardware

Spectrum Interface £15.95
AGF have designed a very cunning interface that allows any Atari/Vic format joystick to be used with the Spectrum. The interface mimics the closure of keyboard switch closures and can thus be used with any software. To use the interface, the keys responsible for controlling the various movement directions are converted to a two digit code according to a programming chart supplied with the interface. This code is then programmed onto the interface by means of two crocodile clips per direction – the clips make contact with a series of bus bars.

The interface was quickly and easily set up and made all of the games packages we tried far more enjoyable by virtue of the ease of play afforded by the use of a joystick.

This interface offers excellent value for money and is a must for any serious Spectrum games player. ▶



PASE

Oric Interface £14.95
The PASE interface is connected to the Oric's printer port and allows two Atari type joysticks to be used with the computer.

The interface is supplied with a short piece of software that is loaded into a spare block of memory at #0400.

Once configured the interface allows the status of either the left of right stick to be interrogated by pieces to locations #400 and #401. The joystick position is returned as a decimal number between 0 and 10.

The interface is more suitable for use within programs written by the user but it should be possible to integrate it within some commercial software.

Comments made above are in respect of the added attraction of using a joystick as opposed to the keyboards in many games program apply equally to this interface.

E&CM

Clares
Providence House
222 Townfields Road
Winsford
Cheshire

AGF Hardware
26 Van Gogh Place
Bognor Regis
West Sussex
PO22 GBY

Mapsoft
Unit 'A'
Oak Road South
Benfleet
Essex
SS7 2BB

Midasonic
9 Firglen Drive
Yateley
Cambridge
GU 17

Kraft Joysticks
SPD Software
Osiers Road
London
SW18 1BR

Cambridge Computing
1 Benson Street
Cambridge
CB4 3QJ

PASE
213/215 Market Street
Hyde
Cheshire

Voltmace
Park Drive
Baldock
Herts
SG7 6EW

Flight Link Control
Unit 12
The Maltings
Turk Street
Alton
Hants

Kempston
180a Bedford Road
Kempston
Bedford
MK4 88L

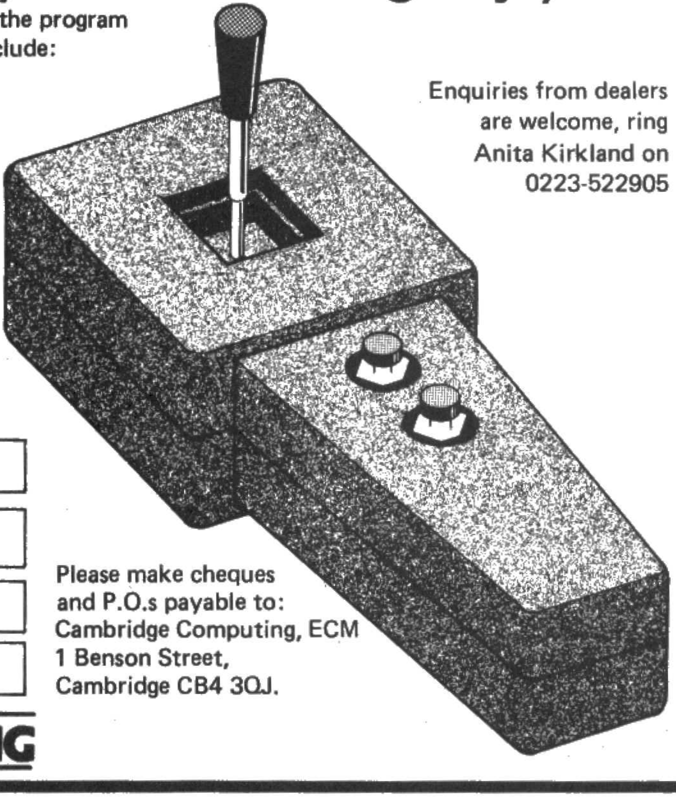
At last! A joystick that works!

Cambridge Computing bring you the first **intelligent** joystick.

Works on all existing software - regardless of which keys the program uses. No need for specially written software. Features include:

- Compatible with Spectrum, ZX81, Jupiter Ace
- 2 Independent Fire Buttons
- 8 Directional Microswitched action
- Plugs into edge connector
- Interface complete with edge connector
- Atari joystick compatible
- Joystick with Interface £29.90

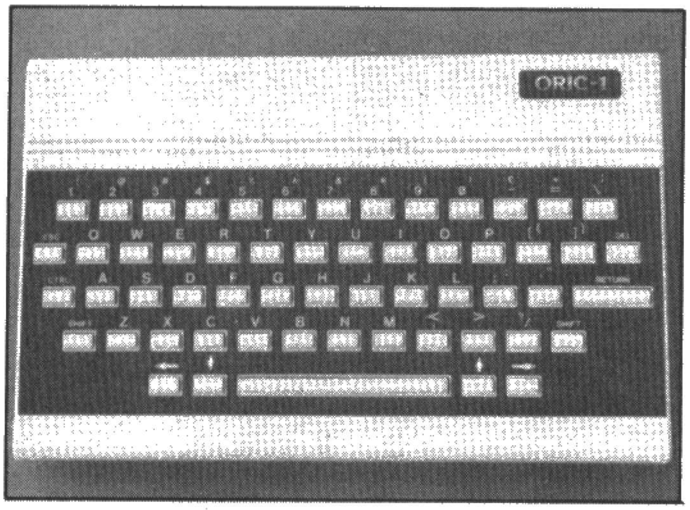
Enquiries from dealers are welcome, ring Anita Kirkland on 0223-522905



Name..
Address..
....
Please send me:
..........joystick, interface, and tape @ £29.90.... £
Spectrum☐ ZX81☐ Jupiter Ace☐
..........interface and tape @ £24.00..................... £
Spectrum☐ ZX81☐ Jupiter Ace☐
..........joysticks @ £7.00.................................. £
Spectrum☐ ZX81☐ Jupiter Ace☐
Total including VAT.......... £

Please make cheques and P.O.s payable to: Cambridge Computing, ECM 1 Benson Street, Cambridge CB4 3QJ.

CAMBRIDGE COMPUTING



THE MUSICAL ORIC

Mike James explores the subtleties of the Oric's sound generator.

One of the first things that everyone discovers about the Oric is that it is LOUD. The best selling Spectrum may have a squeak no bigger than a mouse, and a small one at that but the Oric bellows like a moose! Another difference between these much compared machines is the number of BASIC commands they offer to control the sound that they produce. The Spectrum has a simple but very limited BEEP command to control its single-channel, no noise sound generator. The Oric's sound generator is however, much more sophisticated, with three tone channels and one noise channel, and this is reflected in the number and complexity of its BASIC sound commands – PLAY, SOUND and MUSIC. Although these three commands can hardly take the prize (currently held by BBC BASIC) of being the trickiest sound commands yet invented, the Oric manual leaves a lot to be desired when it comes to explaining how they are to be used. The purpose of this article is to show how logical the three sound commands actually are and to demonstrate how they can be used to produce tailor-made sounds.

PLAY, SOUND and MUSIC

One of the confusing things about Oric sound is that the three commands seem to be too many in that they duplicate each other. This is far from true, as each command has a very clear purpose and use and once this is understood things become easier. The badly named PLAY command doesn't actually produce any sound whatsoever, it merely controls the type of sound that the other two commands, SOUND and MUSIC, will produce. SOUND is the fundamental command that will start the Oric's loudspeaker producing tones or noises. MUSIC is a slightly more refined command that, while only allowing you to control the tone channels, also lets you specify the pitch in terms of the familiar musical scale. To summarise –

PLAY defines the type of sound produced
SOUND produces a tone or noise of a given pitch
MUSIC produces a tone from the musical scale

This means that the procedure in any sound program is first to use the PLAY command to define the type of sound that is to be produced and then to use either SOUND, if a sound effect is needed or the noise channel is to be used, or MUSIC, if a tune is to be played or the noise channel is not required.

The action of the PLAY command is best explained with reference to the MUSIC command because it avoids any complications with the noise channel (which is discussed later). The syntax of the simplest form of the PLAY command is –

```
PLAY channel,0,0,0
```

where 'channel' is a number from 0 to 7 indicating which of the tone channels is 'on' according to **Table 1.**

channel	effect
0	No channels on
1	channel 1 on
2	channel 2 on
3	channels 1 & 2 on
4	channel 3 on
5	channels 1 & 3 on
6	channels 2 & 3 on
7	All three on

TABLE 1

(If you look at **Table 1** carefully it can be seen that 'channel' is simply a three bit number with b0 controlling channel 1, b1 controlling channel 2 and b2 controlling channel 3). Only channels that have been turned on using PLAY can produce any sound although, as we shall see, even channels that are turned off take notice of MUSIC commands. The syntax of the MUSIC command is –

```
MUSIC channel,octave,note,volume
```

where 'channel' is a number between 1 and 3, octave is a number between 0 and 6, and volume is a number between 1 and 15. The MUSIC command will set the specified tone channel to produce a note at a pitch given by 'octave' and 'note' at a volume specified by 'volume'. On the subject of volume, 1 gives the quietest sound, 15 the loudest and most people find that a gentle 3 to 4 is quite loud enough. The way that 'octave' and 'note' work together to define the pitch of the sound is not difficult to understand even for those of you with even a slight knowledge of music.

The western musical scale is divided into 12 notes each a semitone apart – giving the so called 'chromatic scale'. When playing this scale as the top note is reached, then instead of going of into the musical wilderness when the next higher note is played, all that happens is that the scale repeats itself but at a higher pitch – or in other words in the next octave. The fundamental pitch of the scale that the Oric produces is set by the value of 'octave' and the particular note is selected by 'note'. So, for example, the program shown in **Table 2** will play the chromatic scale in each of the octaves that the Oric can produce.

```
10 PLAY 1,0,0,0
20 FOR O=0 TO 6
30 FOR N=1 TO 12
40 MUSIC 1,O,N,4
50 WAIT 30
60 NEXT N
70 NEXT O
```

TABLE 2

Line 10 turns tone channel 1 on and then lines 20 to 70 play each note at each octave. The only other point worth mentioning is that the WAIT command in line 50 is necessary to set the time that each note lasts for. Rather than sounding a note for a specified time the Oric will continue sounding a note until it is instructed to produce another or until all the tone channels are turned off using PLAY 0,0,0,0.

It may seem strange to have to specify the tone channel to be used in both a PLAY command and a MUSIC command. Surely mentioning the channel in just one command would have been enough? The answer is not if you want to play chords! The MUSIC command sets a tone channel to a given frequency and a given volume even if the channel is off so that it cannot be heard. If you want to play a chord so that all the notes start together the obvious way to do it is to switch all the channels off, set the pitch and volume of each one and then turn them all on with a single PLAY command. Try the example shown in **Table 3.**

```
10 MUSIC 1,3,11,5
20 MUSIC 2,4,3,5
30 MUSIC 3,4,7,5
40 PLAY 7,0,0,0
```

TABLE 3

Lines 10 to 30 set up the pitch and volume for three notes and then line 40 switches all three channels on together, producing a beautiful (well harmonic at least) chord.

The SOUND command works in much the same way as the MUSIC command apart from the additional feature of being able to control the noise channel as well as the tone channels. The syntax of the SOUND command is,

```
SOUND channel,pitch,volume
```

where 'channel' is a number between 1 and 6, pitch is a number between 0 and 65536 and 'volume' is a number between 1 and 15. Once again values of 'channel' between 1 and 3 selects one of the tone channels. Now however the pitch of the note is controlled by a single number 'pitch' which gives a high pitched note for 0 and lower pitches as the value increases. Although the command will accept a huge range of values for 'pitch' in practice you will find that values between 1 and 300 are the most useful. Above this range all the notes sound the same! Apart from this difference in the way that the pitch is specified you can use the SOUND command in exactly the same way as the MUSIC command. What sets the SOUND command apart from MUSIC is its ability to control the noise channel.

The Sound Of Noise

The Oric's sound generator has a single noise channel that can be played on its own or mixed with any of the three tone channels. To use the noise channel we first have to turn it on using a slightly extended form of the PLAY command –

```
PLAY tone channels,noise channels,0,0
```

The 'tone channels' part of the command selects a combination of tone channels as before. The new part of the command is 'noise channels' which is a number between 0 and 7 with the same meaning as 'tone channels' except that it selects the combination of channels that the noise source will be mixed with. This idea of mixing the noise channel with the tone channels is the most difficult Oric sound feature to understand. If the noise channel is mixed with a particular tone channel then any sound command that refers to that channel will also produce the noise source and, as we shall see in a moment, any reference to the noise channel will also produce the tone channel. You can deduce from this that to produce a pure tone on a channel you must not have selected it as a channel to be mixed with the noise channel. Likewise if you want to produce pure noise you must have switched off the tone channel that it is being mixed with. Before we can look at some examples, it is necessary to extend the SOUND command so that it can set the 'pitch' of the noise channel. This is quite easy as the syntax of the SOUND command stays the same but the 'channel' number used previously increases its range and becomes 1 to 6. Channel numbers from 1 to 3 set the pitch of the corresponding tone channel but numbers from 4 to 6 set the 'pitch' of the noise channel. For example, if you have used a PLAY command that mixes the noise channel with tone channel 1 then SOUND 4,20,4 will set the noise channel to a pitch of 20 and allow you to hear the noise along with the tone on channel 1 if it is switched on. However SOUND 1,20,4 will set the tone channel to a pitch of 20 and allow you to hear the noise channel and the newly set tone channel if it is switched on. The only problem is that the idea of a noise channel having a 'pitch' is an odd one! In fact the noise channel can produce a 'shhhhhhing' noise at any of 32 pitches. For example, to hear the noise channel on its own at each of its pitches all we have to do is mix it with tone channel 1 and then turn this tone channel of. Try the program shown in **Table 4.**

```
10 PLAY 0,1,0,0
20 FOR I=0 TO 31
30 SOUND 4,I,4
40 WAIT 5
50 NEXT I
60 GOTO 20
```

TABLE 4

Line 10 turns tone channel 1 off and mixes the noise channel with it. Lines 20 to 50 change the pitch of the noise channel and allow you to hear the whoosshing effect produced. To see, or rather hear, the effect of not turning the tone channel off change line 10 to PLAY 1,1,0,0. Now you will hear the same noise channel changing pitch and making the whooshing but you will also hear a tone of constant pitch – this is the tone on channel 1. You can independently change the pitch on both channels without any trouble. To hear this add –

```
35 SOUND 1,31-I,4
```

to the program of **Table 4** along with the change to the play command and notice the head spinning noise that results from the noise channel rising in pitch while the tone channel falls.

The noise channel is the obvious source of most sound effects. For example, the whosshing noise in the last program would make a good rocket take off sound and the EXPLODE sound is clearly based on the noise channel. The two main ways of making the new effects are by manipulating the noise pitch and the noise volume. For example, try the program shown in **Table 5**

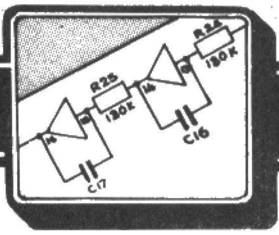
```
10 PLAY 0,1,0,0
20 SOUND 4,2,4
30 WAIT 10
40 SOUND 4,20,8
50 WAIT 5
60 GOTO 20
```

TABLE 5

which produces a noise like a helicopter by sounding noises of different pitch and volume.

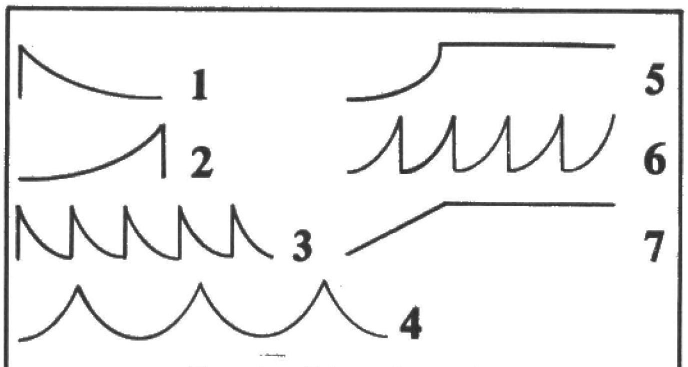
PLAYing an Envelope

The PLAY command has one last surprise in store for us. The last two parameters can be used to select one of seven predefined volume envelopes and durations. This gives the full and final form of the PLAY command as –



PLAY tone channel,noise channel,envelope, envelope duration

A volume envelope is not a difficult idea, it is simply a graph of volume with time but it can be difficult to know what any given envelope sounds like. Fortunately the Oric reduces the choice to a manageable 7 (see **Fig 1**) selected by the value of 'envelope' in the PLAY command. The first two are 'finite length' envelopes, that is they do not change the volume of the note in a periodic manner but have a definite start and a definite end. The first one rises sharply and then declines slowly, the second rises slowly and then declines quickly. To make a sound whose volume is controlled by either of these envelopes all you have to do is use a PLAY command that specifies the envelope and turns on the tone or noise channel in question. Then any SOUND or MUSIC command which specifies a volume of zero will use the specified envelope. For example, to hear the effect of envelope 1 try the program of **Table 6**



The various Oric envelope modes.

```
10 PLAY 1,0,1,100
20 SOUND 1,50,0
30 WAIT 30
40 GOTO 10
```

TABLE 6

Notice that to make the envelope repeat it is necessary to carry out the PLAY command each time. The last parameter in the PLAY command alters the duration of the envelope. In other words it alters how long it takes the note to die away for envelope 1 or how long it takes the note to build up to full volume in envelope 2.

The final five envelopes are periodic, that is they repeat the same overall envelope shape until a PLAY 0,0,0,0 command silences the channel in question. The effect of the envelope duration is much more difficult to gauge for these envelopes in that it alters the repeat rate. Try –

```
10 PLAY 1,0,4,100
20 SOUND 1,80,0
```

which give a fast repeat and then replace line 10 by PLAY 1,0,4,500 which gives a slower rising and falling of volume. It is important to keep in mind that an envelope only alters the volume of a note the pitch is always determined by the SOUND or MUSIC command. Envelopes with the noise channel can be used just as easily.

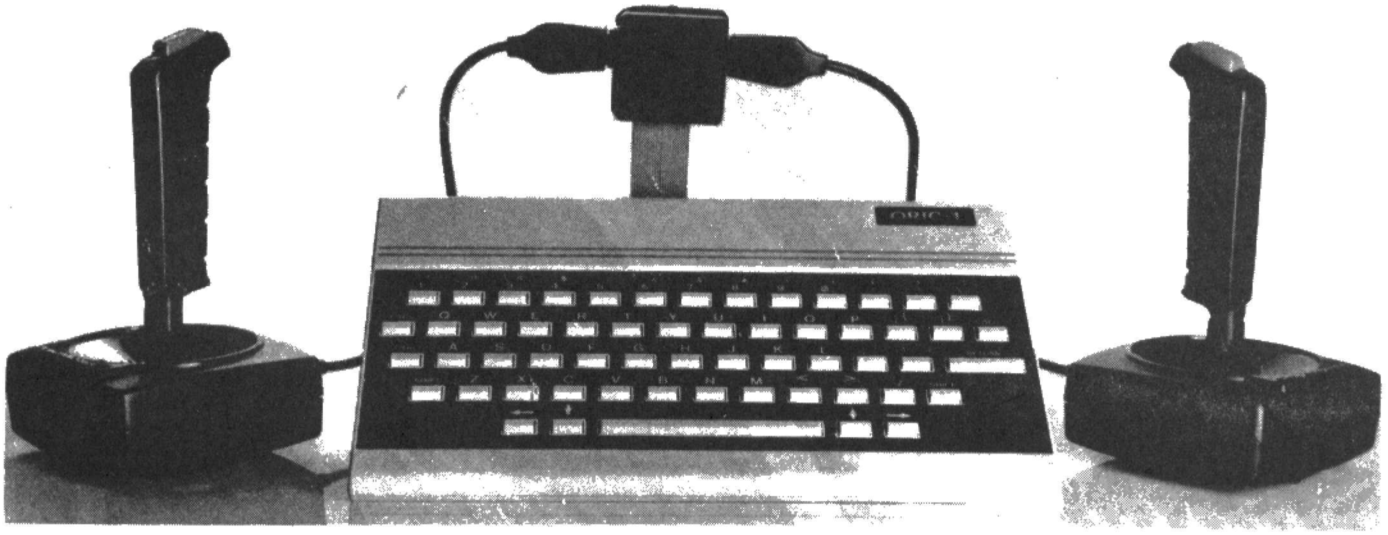
Pitfalls

The Oric's sound commands are very logical and as long as you keep a clear head they are easy enough to use. There are some common mistakes though that are worth looking out for. For example, the keyboard bleep will interfere with any sounds that you have gone to a lot of trouble to produce so it is always a good idea to turn it off with a CTRL and F.

E&CM

P A S E S O F T

ORIC JOYSTICK INTERFACE



ORIC SOFTWARE

● ADVENTURELAND ● WESTQUEST ● WORM ● MACHINE CODE MONITOR
● 5 ALIKE ● DISASSEMBLER **All at £4.99 (inc. p&p)**

213/215 MARKET STREET, HYDE, CHESHIRE SK14 1HF Access/Barclaycard Orders 061-366 5935 24Hrs

Games Programming Techniques

B. Boyde-Shaw concludes that games programs can be educational as well as entertaining in part five of our series 'Understanding Your Computer'.

Computer games can be divided into two distinct categories – namely textual and graphic types. Textual games are probably the easier to program and we'll begin this month with a look at the conception and implementation of a typical adventure type textual game.

Textual

Textual games are those that rely entirely on displaying line after line of instructions, directions and requests for input from the keyboard. The program itself will be full of PRINT and INPUT statements, followed up by IF – THEN clauses to make decisions following the results of inputs from the keyboard.

Usually the player is placed in a given situation, which is quite often straightforward to begin with, but will have snags that are only discovered as the game progresses. For example, a game may have the player wrecked on a beach and having to find a suitable craft with which to escape and food to exist on while looking for weapons for protection from an invading this, that or the other.

All this information would be presented on running the program with a mass of PRINT or similar statements. It would of course be a good idea to have the text easily read and nicely arranged on screen – which quite often it isn't, and not to have too much information presented at once. The program shown in **Table 1** would be a reasonable beginning for the program

TABLE 1

```
 5 CLS
10 PRINT "    YOU ARE ON A DESERTED BEACH"       "2,27"
15 PRINT
20 PRINT "    YOUR CRAFT HAS BEEN WRECKED"       "2,27"
25 PRINT
30 PRINT "    YOU HAVE NO FOOD OR WEAPONS"       "2,27"
35 PRINT
40 PRINT "    BUT YOU HAVE TO ESCAPE SOON"       "2,27"
45 PRINT
50 PRINT "    BEFORE YOU STARVE, ARE KILLED"     "2,28"
55 PRINT
60 PRINT "              OR GO MAD"               "10,9"
```

There are a number of complicated ways of ensuring that all the lines are in the centre of the screen, but the simplest method is to count up the characters used and fill in with spaces, e.g. line 60 – 10 spaces, 9 letters and line 50 – 2 spaces, 28 letters mixed with spaces. Now we wait until the information on the screen is read and we wish to move on. An INKEY$ type command would then be used, to wait for a particular key to be pressed before moving on. Depending on your micro, line 60 could have flashing letters and the whole introduction could be coloured to suit your inclination. The program may be developed as shown in **Table 2.**

So on pressing G to go on, the player is presented with four alternatives, and the skilful use of IF THEN statements etc. puts him further into the program. Choosing West gets him eaten by skarks and the game is over, together with, if your computer can arrange it, suitable sounds and colour changes.

TABLE 2

```
 65 PRINT
 70 PRINT "                PRESS G TO GO ON"            "8,16"
 75 INKEY$ COMMAND (consult your micro manual)

100 CLS
105 PRINT "WHAT IS YOUR NEXT MOVE"
110 PRINT "YOU CAN GO NORTH, SOUTH, EAST OR WEST"
115 PRINT "WEST BEHIND YOU IS THE SHARK INFESTED SEA"
120 PRINT "TYPE, NORTH, SOUTH, EAST OR WEST"
```

Choosing one of the remaining three alternatives would present different situations, only one of which is the one that leads to the escape craft. Choosing the wrong one will place the player in more difficult situations, from which the only way out is retreat.

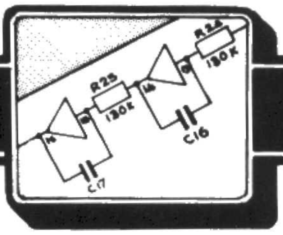
Subtle use of colour and flashing letters can greatly enhance the feeling of dread or imminent danger (but uses more memory) and sounds can be used to good effect. For example, if it is decided to leave the beach and go north (not the correct direction) the screen could immediately go darker as a forest was encountered. Wild natives brandishing spears could then attack the player.

At this point of the program some decisions must be included. One of these must lead out of the situation and some means of retreat must eventually be included, say in this case back to the beach.

The whole idea is to keep the player guessing as to whether he has in fact made the right decision at any point. You can arrange for the player to arrive back at his first port of call if he went the wrong way in the beginning. Well written games of this sort can last indefinitely, but take a long time to plan and program. If you want to write one, start off simple, with lots of REM statements, to guide you through your programs when reviewing it, and leave out all the frills (colour and sound) until you have completed it. Then go to town and dress it up, and try it on your friends on club night, but don't be put off by their remarks, listen and learn. Bought in adventure games of this sort, especially the less expensive ones, can be a little boring if they rely entirely on text.

Adding Interest

So here comes the educational part of the article. Run the program you've bought a couple of times to get the feel of it and make a list of the ways you could make it more interesting. Then list the program (or if you have a printer, print the listing) and decide where to place your improvements. For example, as suggested earlier, line 60 could be made to flash and the whole screen change colour until all the letters used, except the "OR GO MAD" disappear, by colouring the foreground characters the same as the background (PAPER and INK). You could at this point give the player the opportunity to read the introduction once again by having two keys to press, say G and I, either to go on (G) or see the introduction again (I).



Don't forget you and only you can use the improvements; you can't market your improved program.

Games Using Graphics

The most common graphics game is the arcade game, which is normally full of bangs and other disturbing noises and associated very much with death and destruction. Called 'arcade' because these computer games were mainly copies of those games played in 'penny arcades' on large electro-mechanical machines at the drop of a coin, remember?

Nowadays many new forms of graphics games have been designed, some highly sophisticated, some not, but still mostly full of death and destruction. So let's look at our text game in the first part of the article and turn it into a graphics game.

Many new home micros on the market allow you to allocate screen display to part graphics and part text and it is this type of game we will look at.

The text used in lines 10 to 60 could still be printed out in full, or, as on the ORIC with reduced text area, on three lines. Then a picture could be built up using high resolution graphics, showing say yellow sane, a broken boat sinking in the background and the lone figure of a man, preferably made up from more than one character (a multi character block).

The same lines could be used to progress to the next part of the program and then once the initial decision had been made, the graphics scene would change to one of a selection. If north had been chosen, other characters could be introduced throwing, or at least aiming, spears at the player character.

The various scenes could be placed in sub routines or procedures and called on as a result of IF – THEN type statements as the program progresses. Obviously more colour and sound can be used here to good advantage, but would use a lot of memory and to get over this problem, the modern arcade game is written, or at least partly written, in machine code.

Machine code is the language the computer best understands and all Basic statements must be first translated into machine code (or machine language) by the computer's operating system before the computer can carry out the instruction. This, of course, speeds up the operation of the whole program, as the time taken to interpret Basic to the operating language is saved. This means a program can be a whole lot more interesting – that is, more complicated with the extra memory available. It can also be a lot quicker. Machine code is more difficult to write, so again, start off easy and program in Basic. E&CM

48K ORIC 1

Our Price....................164.95

Including P&P AND a free software package entitled "3 Games for Children", which makes imaginative use of Oric's colours, graphics and sound to encourage children to learn counting, spelling and use of typewriter keyboard.

All Orics supplied with DIN to 3 jacks unless DIN-to-DIN specified.

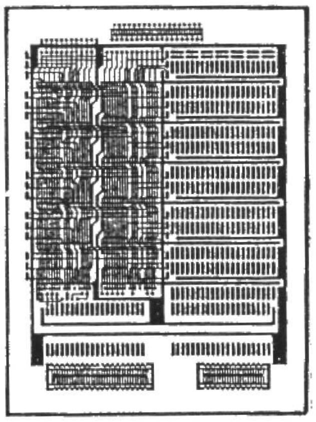
Oric Software

Chess	9.99
Zodiac	9.99
Oric Base	9.99
Oric Flight	7.95
Oric Games	7.95
Teach yourself Oric Basic	6.95
Oric Trek	9.95
Compendium 01	7.95
Backgammon	7.95
3 Games for Children (Cassette and booklet)	7.95
Buy 2 cassettes, deduct	1.00
Buy 3 cassettes, deduct	3.00
Buy 4 cassettes, deduct	6.00

Please add 50 pence P&P for cassette orders.
Cheques/Money Orders etc to:
Burslem Computer Centre
74 Waterloo Road
Burslem, Stoke-on-Trent ST6 3EX
Tel: 0782 825555

ECM08

MICRO COMPUTER DEVELOPMENT BOARD



MCDB-A

FEATURING:
- ★ High quality double-sided printed circuit board
- ★ 100% plated through holes
- ★ Solder resist finish
- ★ Ready laid out inter-connections for memory chips assembly
- ★ Screen printed component side for memory chips
- ★ 3 connector spaces
- ★ Spaces for 12 TTL/CMOS chips
- ★ Spaces for microprocessor and peripheral devices
- ★ Spaces provided for memory pull down resistors
- ★ Dimensions 200mm x 150mm

The Micro Computer Development Board, MCDB-A, accepts up to twelve 24-pin SRAM/ROM/EPROM memory chips with only 3 control pins per chip unconnected. Its unique layout allows any combination of the following memory types to be used: 2016, 6116, 5516, 5517, 2716, 2532, 2732. This list is in no way complete and a number of other memory types can also be fitted.

MCDB-A is ideally suited for most microcomputer developments where static RAM circuits are to be used allowing greater speed, compactness and reliability in microcomputer circuit construction and design.

Price
£14.95 inclusive

WYMER

Available from: **WYMER ELECTRONICS**
8 Briar Place, Dunfermline, Fife, KY11 4BE
Telephone: 0383-418101

ECM08

THE MICROPROCESSOR

In the sixth and final part of our series 'Understanding Digital Electronics', J. Oliver Linton takes a close look inside the microprocessor.

Previous articles in this series have examined the internal architecture of MSI and LSI ICs and showed how they can be used in various circuit configurations. In the final part of the series, we turn our attention to the microprocessor itself.

The diagram shown in **Fig 1** is not intended to represent any particular microprocessor (though it bears more than a passing resemblance to a 6502) and those of you with Z80 or other processors will still find most of the following comments relevant. Before we plunge into the details of data bus buffers and memory address registers, however, let us briefly review what machine code (the language that the microprocessor understands) is all about. (For more detail on this subject, see the series of articles in *E& CM* by Ian Sinclair (June 82 – Feb 83)).

The majority of readers will know that a machine code program is a sequence of 8-bit numbers stored somewhere in the computer's memory. Now each of these numbers may represent one of several things. It could represent a binary number (in the range 0 - 255) or it could represent the ASCII code for a letter. It could be part of the address of a memory location or, most important of all, it could be a machine code instruction. Every microprocessor has a unique code instruction set. For example, to a Z80 the (hexadecimal) number 3E means 'load the accumulator with the number which follows'. To a 6502, however, the same number, 3E, means something quite different – namely 'rotate the specified memory location one bit to the left'. It is not only the code numbers which differ, the two processors have different registers and quite different instructions so it is quite impossible to run programs written for one processor on another, and it is frequently very difficult to translate from one instruction set to another without re-writing the code completely. Nevertheless, the principles upon which all microprocessors operate are essentially the same. They will all have instructions which enable numbers to be fetched from anywhere in the computer's memory into a special register inside the processor known as the **accumulator;** there will be instructions which allow a number of different operations to be carried out on this number (including logical operations, addition and subtraction etc. but not usually multiplication or division); and finally there will be instructions which enable the result to be stored anywhere you like.

But first, the microprocessor must know where in the memory the first instruction is to be found. The address of this location is stored in another very important register in the microprocessor known as the **program counter** and when a piece of machine code is being executed, the microprocessor looks at the number which is stored in the address pointed to by this register. To make this clearer, suppose the section of memory from 0D00 to 0D0A contains the numbers shown in **Fig 2** (all in hexadecimal of course!) and that the program counter contains the number 0D00; the first instruction to be looked at will be A9. Now it is essential that this number be a valid instruction (rather than a piece of data etc.) because the microprocessor cannot distinguish between instructions and data. If by any mischance the microprocessor starts interpreting data numbers as instructions, literally anything can happen and you will be lucky to survive the resulting crash with your program intact. All being well, however, the processor will look at this instruction and carry out its coded messages. Some instructions e.g. 'increment register X' or 'clear carry flag' are complete in themselves and therefore occupy just one byte. Others require two bytes e.g. 'load the accumulator with the number which follows' while others require three bytes e.g. 'store the

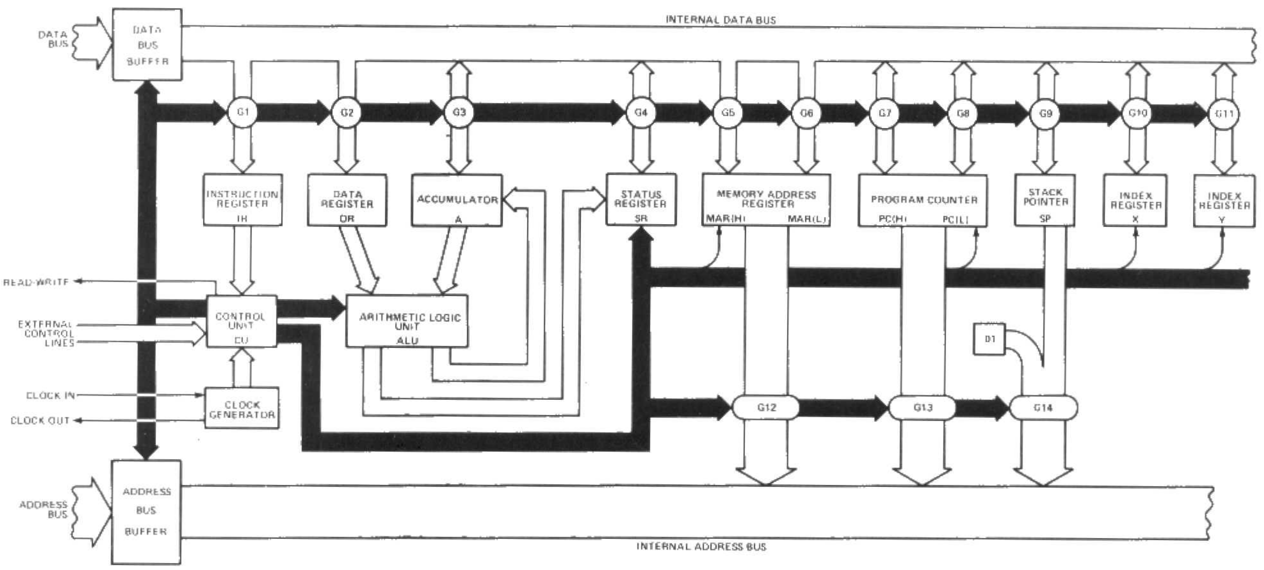


Figure 1. Block diagram of a typical MPU. Although not intended to represent any particular processor it bears more than a passing resemblance to the 6502.

contents of the accumulator in the memory location which follows'. The reason that the latter instruction occupies three bytes (in the case of an 8-bit micro at any rate) is that the address location occupies two bytes not one. In both the 6502 and the Z80 the sequence for a three byte instruction is:

Instruction byte : Low order byte : High order byte

So to store the contents of the accumulator in the location 0D0A, a 6502 would require the 8D, 0A, 0D in that order. (8D is the 'store' command). It is confusing at first to see the address written backwards. The reason is that in some cases it is only necessary to read the low order byte and it simplifies things if the low order byte is always read first.

The Z80 has more internal registers than the 6502 and if anything an even wider diversity of instructions. For this reason, some of the instructions in the Z80 set require as many as four bytes.

When the first number has been read, the microprocessor knows exactly how many data numbers follow it. These numbers are read into the appropriate registers in the microprocessor and the instruction is carried out. At the end of the cycle the processor is ready to fetch the next instruction in the sequence. It will be apparent that each step in the program procedes in two distinct stages. First the appropriate bytes must be fetched from memory; this stage is of variable length depending on how many bytes are required. Secondly the instruction must be executed; likewise this stage is also of variable length depending on the complexity of the instruction. The whole process is known as the fetch-execute cycle.

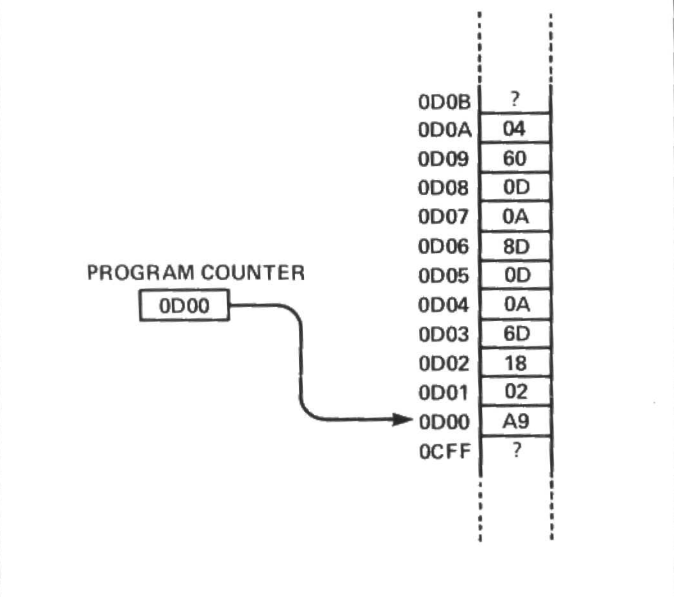


Figure 2. The function of the program counter can be illustrated by considering a section of an MPU's memory.

Now we shall trace the precise effects of the numbers listed in **Fig 2** on all the registers inside our typical microprocessor. **Fig 3** shows our program as it would be written out in 6502 assembler. The first column contains the address where the instruction is stored. It is followed by a mnemonic which helps us remember what the instruction is supposed to do; then the actual numbers which are stored in memory and finally an optional comment. The purpose of the program is simply to add 2 to the number stored in 0D0A (which is assumed for the sake of example to be 04). Before we can begin, it must be assumed that the **program counter** contains the address

```
0D00   LDA   #02     A9 02      : Load accumulator with 02
0D02   CLC           18         : Clear carry flag
0D03   ADC   0D 0A   6D 0A 0D   : Add contents of 0D0A to acc.
0D06   STA   0D 0A   8D 0A 0D   : Store acc. in 0D0A
0D09   RTS           60         : Return
0D0A   ???           04         : (data)
```

Figure 3. A short machine code program

0D00. (Normally this is done automatically by the previous instruction, as we shall see – but clearly special arrangements have to be made for when the computer is first switched on).

The heart of the microprocessor is the **control unit** – and its most important input, the clock, is its pacemaker. The control unit is essentially a highly complex piece of combinational logic which derives its inputs from four sources: a) the clock, b) the instruction register, c) the status register and d) the external control lines. It directs its outputs to virtually every gate and register in the microprocessor. At the start of the fetch stage of the fetch-execute cycle, the output gate of the program counter (G13 in **Fig 1**) is enabled and the address it contains is placed on the address bus. The address bus buffer and the data bus buffer are also enabled and the read/write line set to **read.** A few hundred nanoseconds later, the number stored in location 0D00 (namely A9) appears on the data bus and is read into the **instruction register** (IR) by enabling gate G1. In a typical computer, the clock runs at a frequency of 1 - 2 MHz so one complete clock cycle takes between 500 and 1000 ns. During the second half of this cycle the processor is not idle – the control unit sends a pulse to the program counter causing it to increment by one.

On the next clock cycle, the instruction register contains the hex number A9 which in binary is 10101001. This unique pattern of inputs produces a unique set of outputs from the control unit which enables the output gate from the program counter (G13) once again and the input gate to the **accumulator** (G3). In an exactly similar manner this causes the data contained in the next location (that is the number 02) to be read into the accumulator. During the second half of the cycle, the program counter is incremented yet again ready for the next instruction. Two clock cycles have passed and the instruction is completed.

The next instruction (CLC or clear carry flag) is only one byte long but it too takes two clock cycles. The first cycle is always the same and results in the number 18 being deposited in the instruction register and the program counter being incremented (this incidentally now contains the number 0D03). On the second cycle, the control unit clears the carry bit in the register known as the **status register.** (The various bits in this register are set automatically by the **arithmetic logic unit** and indicate for example whether or not the result of a calculation was positive or zero or whether a carry was generated etc. In the 6502 it is always necessary to clear the carry bit before performing an addition).

The next instruction, in location 0D03 is a three-byte instruction which takes the number in location 0D0A and adds it to the accumulator leaving the result in the accumulator. It takes 4 clock cycles to complete. Its operation is summarised in **Table 1.** (The read/write line may be assumed to be at read unless stated otherwise).

It is worth noting that the program counter is incremented during the second half of every cycle of the FETCH stage, but not of course during the EXECUTE stage.

The instruction which follows (STA 0D0A, store contents of accumulator in location 0D0A) is very similar to the previous one. In ▶

TABLE 1

Cycle	Operation	Result	
FETCH STAGE			
Cycle 1a	Enable program counter output (G13) Enable instruction register input (G1) Enable address and data bus buffers	IR	= 6D
1b	Increment program counter	PC	= 0D04
Cycle 2a	Enable program counter output (G13) Enable memory address register (low) (G6) Enable address and data bus buffers	MAR(L)	= 0A
2b	Increment program counter	PC	= 0D05
Cycle 3a	Enable program counter output (G13) Enable memory address register (high) (G5) Enable address and data bus buffers	MAR(H)	= 0D
3b	Increment program counter	PC	= 0D06
EXECUTE STAGE			
Cycle 4a	Enable memory address register output (G12) Enable data register input (G2) Enable address and data bus buffers	DR	= 04
4b	Send signals to arithmetic logic unit causing it to add together the numbers in the data register and the accumulator, depositing the result in the accumulator and setting flags in the status register accordingly.	A	= 06

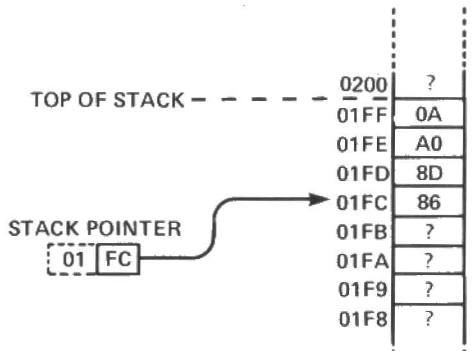


Figure 4. The action of the stack pointer is illustrated by this representation of the stack and the instructions listed in Table 3.

TABLE 2

Cycle	Operation	Result	
•EXECUTE STAGE			
Cycle 4a	Enable memory address register output (G12) Enable accumulator output (G3) Set read/write to write Enable address and data bus buffers	0D0A	= 06
4b	Does nothing		

TABLE 3

Cycle	Operation	Result	
FETCH STAGE			
Cycle 1a	Enable program counter output (G13) Enable instruction register input (G1) Enable address and data bus buffers	IR	= 60
1b	Increment program counter	PC	= 0D0A
EXECUTE STAGE			
Cycle 2a	Enable stack pointer output (G14) Enable program counter (low) input (G8) Enable address and data bus buffers	PC(L)	= 86
2b	Increment stack pointer	SP	= FD
Cycle 3a	Enable stack pointer output (G14) Enable program counter (high) input (G7) Enable address and data bus buffers	PC(H)	= 8D
3b	Increment stack pointer	SP	= FE

fact the FETCH stage is identical. The EXECUTE stage is summarised in **Table 2.**

The final instruction is much more complex than it looks because in spite of occupying only one byte, it takes 6 cycles to complete. The reason is that the RTS instruction causes the microprocessor to jump back to the place from whence it came just like 'RETURN' in BASIC. The return addresses are stored in a special area of memory known as the STACK. There is also a special register inside the microprocessor known as the **stack pointer register** which points to the current location on the stack. The 6502 expects to find its stack between 0100 and 01FF and therefore only requires an 8-bit register – the first 8 bits on the 16-bit address being set to 00000001. (With the Z80, the stack can be placed anywhere in memory and accordingly a 16-bit register is provided). Whenever a jump to a subroutine is executed, the contents of the program counter are pushed onto the stack, high order byte first, then low order. Whenever a number is stored on the stack, the stack pointer register is decremented by one, and whenever a number is retrieved, the stack pointer must be incremented by one. In this way the pointer is always pointing to the **last** entry to the stack, and hence this number will be the **first** to come out. So the microprocessor keeps track of all the subroutine addresses to any depth – provided of course there is enough memory space on the stack.

Let us assume, for the sake of example that on executing the RTS instruction, the stack pointer contains the number FC and the relevant area of memory is as shown in **Fig 4.**

What happens now is shown in **Table 3.**

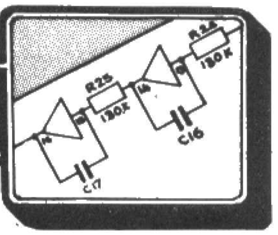
As to why this instruction takes the 6502 six cycles to complete and not three I have to confess ignorance! A table showing the contents of all the internal registers during the execution of this program is shown in **Fig 5.**

Two registers shown in **Fig 1** have not yet been mentioned, namely the X and Y index registers. These registers can be used as temporary stores and are frequently used to index several items in a list. For example, the instruction:

```
LDA    0D0A,X    BD 0A 0D
```

loads the accumulator not with the contents of location 0D0A as before, but with the contents of location 0D0A + X, where X is the contents of the X register. In order to do this, the low order byte of the address is placed in the data register and the contents of the X register is added to it before it is placed in the low order byte of the memory address register. If a carry is generated by this addition, the high order byte of the memory address register must be incremented by one.

Other instructions, not described above include shift and rotate operations, comparisons, conditional and unconditional jumps and many more. Some of these operations require more registers than those shown in **Fig 1** but I think enough has been said to indicate both the complexity and the logic behind the workings of a microprocessor. One's admiration must, however, be reserved for the control unit into



	IR	DR	A	SR	MAR H	MAR L	PC H	PC L	SP	X	Y
LDA #Ø2	A9	–	Ø2	Ø	–	–	ØD	Ø2	FC	–	–
CLC	18	–	Ø2	Ø	–	–	ØD	Ø3	FC	–	–
ADC ØD ØA	6D	Ø4	Ø6	Ø	ØD	ØA	ØD	Ø6	FC	–	–
STA ØD ØA	8D	Ø4	Ø6	Ø	ØD	ØA	ØD	Ø9	FC	–	–
RTS	6Ø	Ø4	Ø6	Ø	ØD	ØA	8D	86	FE	–	–

which all these instructions are pre-programmed. In larger computers the control unit is itself a tiny microprocessor executing its own microprogram in its own microlanguage – but to pursue that road here would only beg the question – how does the control unit work? In a conventional microprocessor, the control unit is nothing more than a collection of gates with a lot of inputs and lot of outputs and the design of a new microprocessor is largely a matter of designing the logic of the cintrol unit so that it performs the required functions. It is no accident, therefore, that all the 6502 codes of the form xxx11001 are immediate (2-byte) instructions, or that most of the Z80 instructions ending in 111 concern the accumulator. These consistencies simplify the design of the control unit which in any case requires the use of large computers. The second aspect of the design of a microprocessor is the implementation of the final logic design onto a chip of silicon. In this process too, computers play a large part. One is reminded of the way in which first generation computers were powered up from cold. First a tiny program a few words long was entered into the computer by physically setting certain switches. The function of this program was to read in a slightly longer program from paper tape which enabled the computer to read an even longer program from magnetic tape . . . Today we see small computers helping to design larger (or rather, smaller but even more powerful) computers which help to design even more powerful computers . . . Will the process ever end? Not in the near future at any rate. What is certain is that, for the forseeable future the fundamental building blocks of the computer will be the humble NAND gate and the S-R bistable, and if you understand how these components work and how they can be put together, you can truthfully say that you know 'how a computer works'. I hope that this series has helped you, the reader, some way along this road and opened your eyes to the true wonder and fascination of the digital electronic computer.

E&CM

ZX81 OWNERS

SPECIALISED PRODUCTS
MODULAR EASY TO USE
FOR HOME/INDUSTRY/EDUCATION

TE10 INPUT/OUTPUT PORT — Easy to use. Fits between ZX & RAM PACK/PRINTER (if required). No skill required to connect. Can be used for such things as:- motor control; sound/music generators, connection to printers/floppy discs/light/pens/other computers, temperature monitoring, square wave generating, control of rotating aerials, even train sets etc. Port has 16 input/output and may be used without any electronics knowledge to connect other add-ons listed below. Motherboard required ONLY when two or more add-ons are used at any one time.
FULL ASSEMBLED **£17.95** KIT (without case) **£14.95**

TE12 4 CHANNEL RELAY BOX — Plugs in to TE10 Contact rating:- 240v AC/1.5A – 24v DC or 110vAC/3A. Up to 4 units i.e. 16 relays can be operated – **£14.95**

TE15 8 WAY TRANSISTOR DRIVER — **£9.95**

TE17 8 WAY SWITCH UNIT (EDUCATIONAL) — **£12.95**

TE18 8 WAY INDICATOR UNIT (EDUCATIONAL) — **£12.95**

TE20 JOYSTICK & FREE GAME — Plugs in to TE10 (2 Joysticks may be connected via Motherboard)

TE30 MOTHERBOARD — Allows multiples/combinations of add-ons — up to 16 I/O lines may be used —**£15.95**

TE40 ANALOG TO DIGITAL UNIT — 8 channel for use with I/O Port — **£24.95**

TE126 POWER SUPPLY — Required for use with add-ons TE12/15 & 18 — **£4.95**

23 + 23 WAY ZX Edge Con. — **£2.85** Contact Cleaner — **£2.30**

16 WAY SINGLE SIDED EDGE CON. — **£1.95**

EXTENDED PIO NOTES — **£1.00**

ALL PRICES INCLUDE VAT

Receipts always provided — Delivery normally ex-stock. ADD 50p towards p&p on all orders under £20.00, with the exception of accessories, e.g. Edge con. Full instructions and examples with all products.

ECM08

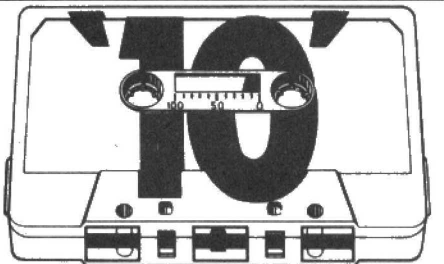
SEND S.A.E. FOR CATALOGUE
POSTAL OR TELEPHONE ORDERS ACCEPTED

VISA

THURNALL (ELECTRONICS) ENG.
DEPT. E, 95 LIVERPOOL ROAD,
CADISHEAD, MANCHESTER M30 5BG
TEL: 061-775 4461

STARTECH

Absolutely the lowest prices!



BLANK DATA CASSETTES

£5-95 Inc VAT Postage and Packing 45p

Computer Grade Data Cassettes suitable for use in all of todays home computers. Each comes with its own case.

Tick for further information
☐ VIC 20 PRODUCTS
☐ COMMODORE 64
All prices subject to availability or change without notice.

PLEASE SEND ME
Pack of 10 Data Cassettes

Qty	Price	TOTAL
	£6.40 inc. p&p	

Name ______________________
Address ______________________
____
I enclose Cheque, P/O for ______________________
CREDIT CARD ☐ ACCESS ☐ BARCLAY CARD ☐ (24hr. ANSAPHONE SERVICE)
☐☐☐☐ ☐☐☐ ☐☐☐ ☐☐☐

EC1

STARTECH 208 Aigburth Rd, Aigburth, Liverpool L17. 051 727-7267

Open letter to the Micro Industry

We feel that we should warn any member of the public who intends to purchase an RS232 interface for use with a printer and a Sinclair ZX Spectrum. We bought the Sinclair Spectrum RS232 c Interface from a well known company and waited 12 weeks for it to arrive, even though our cheque as cashed immediately. This is shoddy business practise and illegal. More seriously, despite what it states on the packaging, the Interface will NOT allow you to use LPrint or LList – surely the main reason for purchasing such a product. Despite a personal visit by our staff to the Company's premises, and very many telephone calls, we have been continually fobbed off with devious and misleading statements, and have not received a refund or any apology. We have been in the micro business a little longer than most, and see this sort of treatment as the worst form of arrogance, cynical advertising, and rotten treatment of customers.

It is behaviour like this that harms all of us, and we see it time and time again, from the biggest in the land to the back street cowboys. To all of you in the micro industry – you must not advertise goods that are not readily available, you must not stick money in the bank before you despatch the goods, you must never mislead the customer by lying or making stupid claims in your advertising. We do our best to follow these ground rules the chances of all other companies doing the same seem remote.

Automata UK Ltd
Portsmouth
England

Calling Technical Software Authors

Electronics and Computing Monthly are planning a new editorial review section for really 'smart' software. This regular feature will be an opportunity for software authors to reveal details of their specialist scientific, engineering or other technical software packages which they believe may have applications for use by other *E& CM* readers.

While these software ideas should not be too esoteric, Electronics and Computing does have an audience which is just a cut-above the average computer enthusiast and which includes quite a large number of electronics and computing design engineers, scientists and technicians, plus, of course, very clever enthusiasts who enjoy their hobby at a distinctly high academic level.

Our plan is to publish reviews of selected software describing its applications and technical specifications. Readers who require more information or would like to discuss swaps or purchases of software will be put in touch with the relevant authors.

As a first step write down some operational details for your creations and send us a short synopsis (approximately 250 words maximum). Please do not send disks, tapes or listings at this stage until we have had chance to assess the programs suitably.

Write to:
Technical Software Editor
Electronics and Computing Monthly
Scriptor Court
155 Farringdon Road
London EC1R 3AD

Spectrum Vectors

Dear Mr. Evans,

The Spectrum printer interface you published in your July issue appears to be below your usual standard. I am sure that the author is a more proficient programmer than myself usually, but in this particular case he has appeared to miss the fact that the Spectrum **has** got a printer driver vector.

I have produced a printer driver program that uses the Kempston parallel port, which I seemed to remember costs about £18, (not their printer driver). It consists of a 15 line BASIC program that is run after the computer is switched on. Thereafter, LPRINT and LLIST work via an Epson MX80 FT2. Chunky graphics are printed, and the user graphics are printed as their corresponding letters, and the £ prints although it is in the # position in my printer. (The reason why I can still use # is that it isn't changed in italics mode).

The relocatable machine code routine that is given as DATA in the BASIC loader is only 53 instructions long plus 10 entries in a re-locatable table to correalate the standard graphics with the printer. **There is a checksum at the end of the data entry so that the program can be typed in and it self checks when loading.** This should be done more often in programs presented for the user to type in.

I have been offering this for sale for some months for £1, and 50 copies for £15 to clubs. The program is presented as the BASIC listing, an A4 sheet containing about 90 lines of notes, and a further A4 sheet containing the annotated assembly listing.

John de Rivaz
West Towan House
Porthtowan
Truro, Cornwall

Owning Up

A number of points concerning recent articles have come to light.

May 83
Spectrum Printer Interface
Optional line-feed does not work with lines of 72 or more characters. To correct this substitute 28 for 18 at address FEAC in **Fig. 70.**

June 83
Power control for micros
Some people have had difficulty in locating a supply of Stepper Motors.
McLennan Servo Supplies
Yorktown Industrial Estate
Doman Road
Camberley
Surrey
can supply a suitable motor, the Impex 35, at a cost of about £10.00.

Spectrum Printer Interface
Fig 3. All 5V connections go to the output of voltage regulator IC7, not to the edge connector as implied.
Fig 3. All connections to chassis are in fact to 0V.
Fig 3 (again). IC6, pins 9-17 (bar pin 12) should go to D0 through D7 of the edge connector.

We read with interest the letter written by Messrs Gibbons and Hill, recently published in your Letters Column, regarding the formation of a 6809 newsletter. We would like to draw your readers' attention to the fact that there already exists a specialist user group in this country, specifically geared to meet the needs of the 68 microprocessor user in Great Britain and Europe.

Please – there are so few 68 microprocessor users around – lets stick together!

Yours sincerely.
Jim Anderson
for 68 MICRO GROUP
41 Pebworth Road
Harrow
Middlesex HA1 3UD

Oric Software Reviewed

Nick Clare has put a range of Oric software through its paces... here's his report.

It's widely recognised that for any home computer to achieve sales it must be supported by the trade both with hardware add-ons and, more significantly, by a range of software that is both as large and diverse as possible. Production of the Oric, after a slow start, is now at a level that is starting to meet the demand for the computer and to coincide with this, a number of companies have completed development of their first batch of software for the machine and are starting to market their effort.

The Oric is based on the popular 6502 MPU, a micro that is familiar to many machine code programmers. This has meant that most software houses have only had toi familiarise themselves with the idiosyncrasies of the Oric itself rather than embark on a lengthy familiarisation of a new MPU. The overall result of this is that the quality of software offered for the Oric is, in general, higher than might be expected from the first batch of software from a new computer.

No Problem

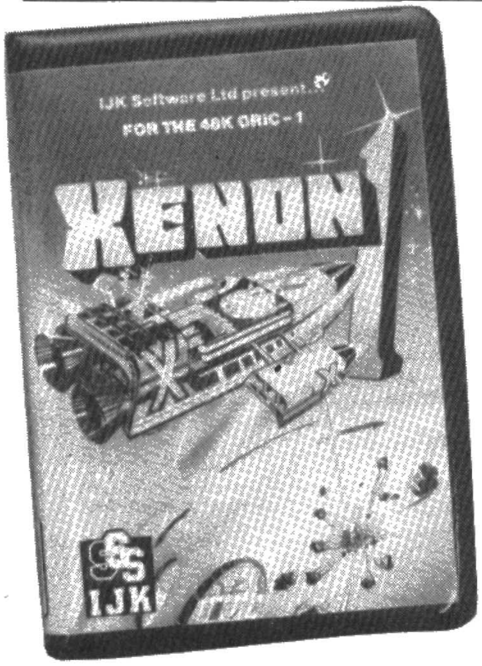
The Oric has attracted a great deal of comment as to the reliability or otherwise of its cassette loading system. The computer offers two loading speeds, the default rate being 2400 baud. This is eight times faster than the machines other transfer rate of 300 baud – the speed at which, for example, **CUTS** are recorded. Clearly, this fast speed is squeezing the most out of both the cassette recorder and the Oric's Software/Hardware signal processing system. Most of the software reviewed for this feature provided two recordings of the Program, one at the fast speed and another at the slow speed – some manufacturers played it very safe and provided only a slow version of their program.

Having mentioned the potential problems associated with the fast loading speed, it must be said that, with the majority of the software offered at 2400 baud, loading tool place without any problems. Getting the volume level right was obviously important and the tip here seems to be, start with a high volume setting and gradually reduce the setting until the loading is reliable – this in contrast with the advice given for most machines that suggest increasing volume until data transfer is reliable. The Oric's cassette recording system is also free from the saving and loading circuitry that affects, for example, the Spectrum – no need to unplug the mic lead when loading programs. Indeed, the confidence in this aspect of the design is shown by the provision of a DIN recording lead as standard.

The only other thing to emerge from our time with the Oric, of interest to those of you contemplating using the machine's RGB output, is that the machine produces positive video signals and negative sync.

How Subjective?

At the risk of pointing out the obvious, it's probably worth stating that any review of a game must reflect the personal bias of the reviewer. Whilst attempting to set out as an objective as possible assessment of the packages for the more mentally stimulating rather than action orientated programs is likely to slightly bias each review. In the case of the more 'serious' programs, personal tastes are again bound to break through – what's a perfect assembler to you may be an inelegant package in the eyes of someone else. Having laid bare any personal bias, hopefully you'll be able to weigh any comments according to your personal tastes.



IJK Software
3D Maze and Breakout
48K £7.50

This package from IJK offers two excellent games for the price of one.

The first, 3D Breakout autostarts and begins with the information that the aim of the game is to pilot your way through a maze. The maze's exit is at the South East corner, while play begins at the North West corner. The player is asked to input the size of the maze, from 3 to 20 squares, at the start of the game. In the case of the larger sizes which require some time for the random generation of the maze, a message explaining the delay together with a 'countdown' is produced.

Play begins with a 'bird's eye' view of the maze with the player's position marked together with the complete topography of the computer generated maze. This display lasts for only a few seconds during which time the layout must be memorised.

The global plan of the maze is replaced by an impressive graphic representation of an eye level view from within the maze. The screen displays the direction in which the player is facing as well as instructions on how to move and to change orientation. The < and > are used to change direction which the L key is ued to move forward through the maze. Each change of direction or move is accompanied by a 'rushing' sound effect.

The ultimate object of the game is to escape from the maze but the computer keeps a count of both the number of moves made during the escape and of the time the computer was waiting for you to make a move.

If the player gets hopelessly lost during the escape it is possible to display the global view of the maze but, at a cost of 20 moves.

This game was very enjoyable to play and made good use of the Oric's sound and graphic capabilities. It would be unlikely to become boring as each time it is played a new challenge is presented – very few people would be able to master the 20 x 20 grid.

Breakout is a version of the familiar arcade game. The game from IJK offers a number of variations on the standard game, the player being able to select a single wall, a double wall with narrow gap or a double wall. The game can either be configured to automatically increase the difficulty of the games as each complete wall is destroyed or to select the required bat size (from large to very small) and ball speed.

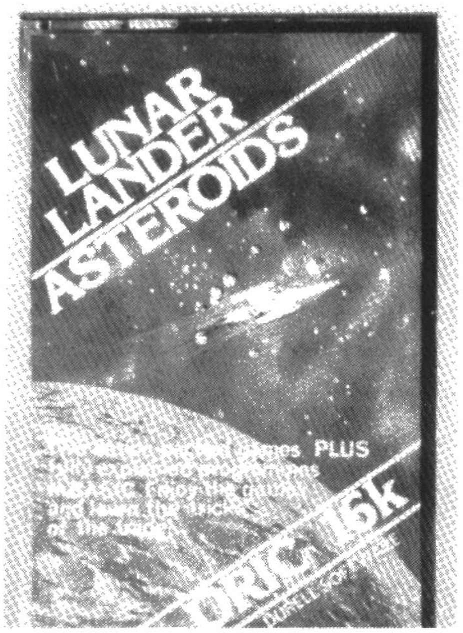
All this gives 10 skill levels to cope with the demoands of anybody from the Breakout tyro to the expert player. Again this game makes good use of the Oric's graphic and sound facilities.

The cassette offers good value for money, offering two excellent yet very different games.

Also received from IJK – X Enon 1, an arcade type space invaders game and Candy Flott and Hangman, two programs aimed at younger players with an emphasis on education in addition to entertainment.

Durell Software
Lunar Lander & Asteroids
16K £6.95

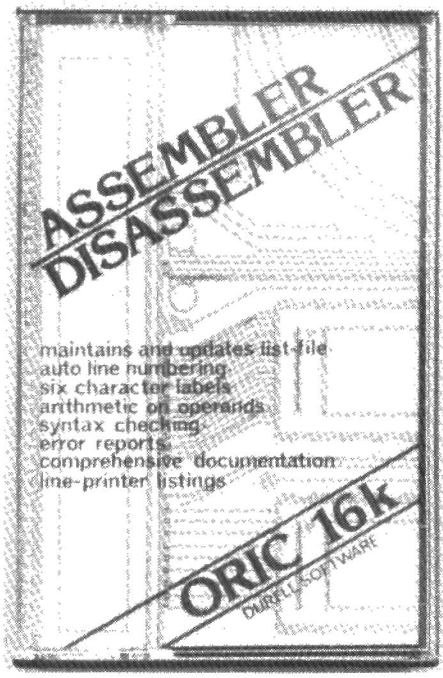
An unusual package in that the excessive jealousy with which most software houses guard their programs is replaced with an invitation to examine the listings and learn



the 'tricks of the trade'. Each side of the tape features two versions of the game – the first auto starts going straight into the action while the second prefaces each line of the program with a REM statement. The latter version of the game does not run automatically and the user is free to list the program, using the space bar to halt the output at any stage.

Both games are fairly simple arcade type simulation with Lunar Lander presenting the player with a Lunar Module about to start its descent from the top of the screen. The numeric keys from 0 - 9 are used to adjust the rocket's thrust – the aim of the game being to land the ship at a velocity between 0 and — —10 ft/sec. A range of on screen displays show the velocity, height, thrust and fuel status of the module.

Asteroids is again a fairly simple game in which, by means of thee keyboard controls (left, right and hold) a space ship must be piloted through an asteroid storm.



This tape offers two fairly basic games but the major attraction must be that each game may be listed and the various techniques used examined for inclusion in the user's own programs.

The second tape from Durell was an assembler/disassembler which, offers auto line mumbering, six character labels, arithmetic on operands, syntax checking, error reports and line-printer listings. The program is supported by comprehensive documentation and should offer the assembly language programmer most of the facilities required for this field of endeavour.

PASE
Machine Code Monitor
16/48K £4.99

The machine code monitor allows the programmer to utilise the full power of the 6502 processor. The package is menu driven and provides the following options.

H allows a new value of **HIMEN** to be set.

X to set the start address of displayed memory.

W displays 72 bytes of memory in a 12 row by 6 column format. In this mode, a window may be positioned over any location by means of the cursor keys and the contents of the selected altered to a new value. Pressing the space bar will move the display to the next 72 bytes.

T will transfer a specialised block of memory to another area of memory.

R will display the current state of the 6502's register.

C enters a call address.

S allows a routine to be saved on tape.

L loads a saved routine.

B returns control to Basic.

M returns the user to the men page at any point.

A useful package for the machine code programmer although it should be noted that all programming is done in decimal or hex op-codes and not assembly language statement.

Also from PASE – Worm, an action game with farily basic graphics although an entertaining experience; 5 – alike, Adventureland, Pioneer 1847 and another 6502 disassembler. All programs retail at £4.99.

Crescent Software
Typical
16K £5

Typical is a touch typing course for the Oric. The program is menu driven with the options, in addition to a set of instructions, being an introduction to the home keys, the G & H keys, T & Y keys, the top row keys, the bottom row keys, numbers and a general practice session.

In each section of the program, 'words' consisting of the chosen letter groupings are randomly generated and the trainee invited to enter the groupings using the approved fingers. A display at the top of the screen indicates the position of the next key to be pressed on the keyboard.

At the end of entering text, the program calculates both the speed of typing and the accuracy. The user is then advised to either try the same 'words' again or to move on to a different set of 'words' from the same letter grouping.

The practice section of the program allows the user to enter his own text and to copy this via the typing tutor.

Typical is supplied with a comprehensive manual that, not only introduces the disciplines of touch typing but also aims to allow even the "micro-novice" to load and run the program to good effect.

If the course is followed it should provide as good an introduction to touch typing as any and be better than most such courses. The automatic marking of each exercise is a very valuable feature.

Severn Software
Jogger
48K £6.95

Jogger follows the pattern of many other programs by offering both a slow and fast version of the game. The fast version loaded reliably in about 3½ minutes and autoran with a message asking whether or not the game's instructions were required.

The object of the game is to steer a jogger who must be very keen indeed to achieve the fitter leaver look across four lanes of motorway traffic, to the other side of a crocodile infested river to the safety of a box at the top of the screen. The game was reasonably enjoyable to play although the screen was rather crowded meaning that the resolution of each element of the display was rather crude. The game offered a number of skill levels meaning that it could continue to provide a challenging game for some time.

Two other games from Severn, again both for the 48K Oric, were Moria and Grail. These are both quite similar adventure type games with allusions to Lord Of The Rings and the Monty Python escapades respectively.

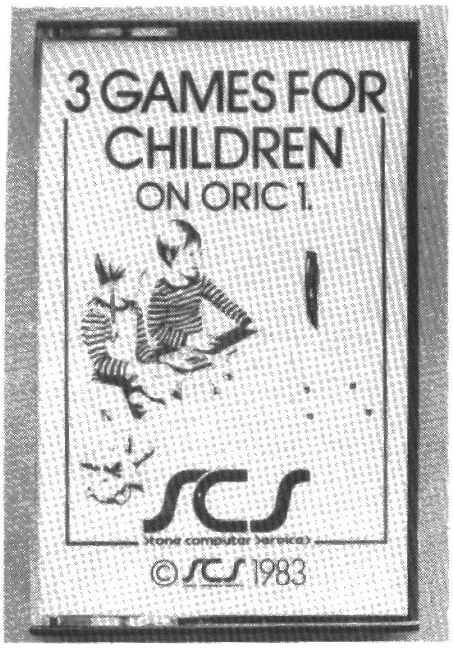
Crunch Computer Systems
Crunch Math
48K £7.50

Crunch sent us three pre-production tapes of which we chose Crunch Maths for a closer look. Following the pattern of other tapes, the program was split into two parts – the first a detailed explanation of the program, the second being the program itself.

Upon loading the instruction section of the tape, the user is given the option of dumping the text to a printer or of scrolling the output on the screen.

Crunch Math is a package designed to augment the facilities of the Oric's floating point package and for any one who desires to use the computer to undertake any scientific work, the package should prove a useful tool.

The other programs from Crunch were a Disassembler and Go-trainer (both £7.50). ▶



A & F Software
Zodiac £6.90

Of the two adventure games from A & F (Death Satellite and Zodiac) we tried our hand at the second. These are text based packages with little of the sound or graphic potential of the Oric being called upon. Having said that though, the games looked as if they could be fascinating to play if plenty of time were available.

The instructions are very sparse with only four commands explained and the information that there are many more required to play the game but that these must be learnt as the game progresses.

Our game started by picking up a number of seemingly useful items but ended when we entered a cave and were told it was too dark to continue. One of the items picked up was an oily rag on a stick and we'd passed a burning building. Light the rag in the fire and things would be hunky dory. Unfortunately quite how to accomplish this we could not work out in the time alloted to review the game.

The feeling is though, that the game would be worth returning to in more relaxed times in order to further explore the game.
Also available – Oric Painter at £6.90.

Elephant Software
Jerico 2 £6.95

Elephant sent us a pre-production version of their game Jerico. The game involves the player in managing an army that is attempting to bring down the famous walls of Jerico. There is plenty of action with the player responsible for keeping his troops happy and fed – thus keeping them loyal – as well as directing various raids to weaken the city's walls and to gather arms.

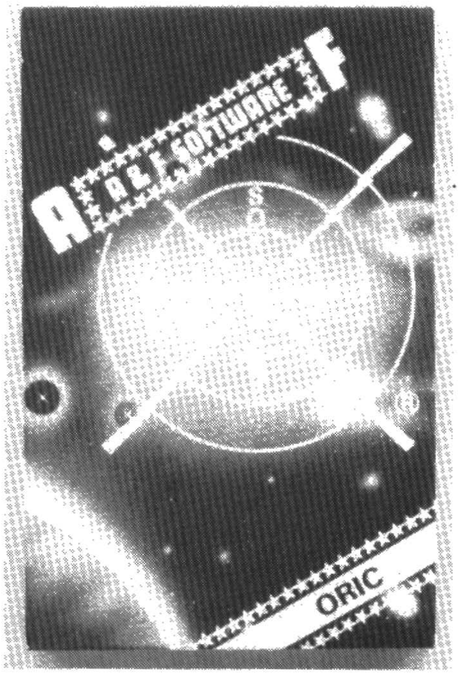
If you've a taste for power, this game should be right up your street.

Another Oric game supplied by Elephant is Vanquisher which retails at £6.50.

Stone Computer Services
3 games for children £7.95

This is one of the few tapes we had problems loading, although it was not a problem with the Oric but rather a cassette housing that was reluctant to allow the tape to move. Loosening a couple of screws cured the problem and, although this was in all probability a one off problem, it does highlight the need for adequate quality control unit with 100% testing of all tapes being the only way to achieve this.

The games themselves are, as the cassettes' titles would suggest, aimed at young children. The laudible aim is to familiarise children with the computer by way of a counting exercise that involves identifying and counting a variety of shapes; a colour and spelling test that asks for the colours of various shapes to be spelt out a correct letter entry being rewarded with the Oric's Ping sound while any mispelling produces the Shot sound.



The final game is a typing test that displays the alphabet and numerics as double height characters.

For the very young, this tape represents an ideal way of both learning and, at the same time, overcoming any dormant computer phobia.

PSS
Centipede

A version of the popular arcade game. The player must destroy a fast moving Centipede in a game that closely resembles a space invader game.

The players base may be moved to the left and right as well as a limited vertical movement.

The game was fast and reasonably enjoyable.

Quark Data
Flight Simulator

An Oric version of the game that puts the player in the pilot's seat of a jet airliner. The plane must be put on the correct course and then using a variety of throttle and flap controls, brought to a safe landing.

The game makes good use of the Oric's graphics capability (the instrument display is very realistic) and the machines sound generator.

Microplot
Editor Assembler/Disassembler 48K

A Basic program that provides a powerful machine code handling facility. The program supports the full 6502 assembly code plus labels with auto branch calculation, and psuedo ops.

Programs can be loaded with the Oric's CLOAD command for disassembly or editing and similarly, edited programs may be saved.

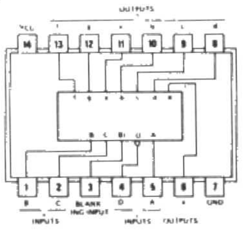
We'll return to this package at a later stage when we deal with Oric machine code programming in greater depth.

IJK Software 9 King Street Blackpool Lancs.	**Crescent Software 3 Rolfe Crescent Heacham Norfolk PE31 7SF**	**Elephant Software 41 Haymill Road Burnham Berkshire SL1 6NE**
PASE 213/215 Market Street Hyde Cheshire SK14 1HF	**Severn Software 5 School Crescent Lydney Gloucestershire GL15 5TA**	**PSS 452 Stoney Stanton Road Coventry**
Stone Computer Services 27 Mount Crescent Stone Staffordshire ST15 8LR	**A & F Software 830 Hyde Road Manchester M18 7JD**	**Quark Data PO Box 61 Swindon Wiltshire**
Durell Software Higher Combe Combe Florey Taunton Somerset TA4 3JF	**Crunch Computer Systems Ltd 78 Victoria Road Swindon Wiltshire**	**Microplot 19 The Earls Croft Cheylesmore Coventry**

INTEGRATED DATA

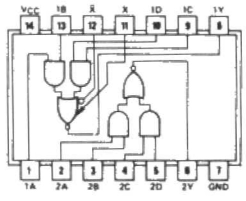
More pin-outs for 74xx TTL ICs

49



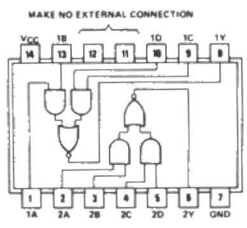
BCD-TO-SEVEN-SEGMENT DECODERS /DRIVERS

50



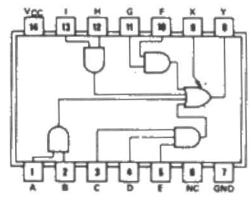
DUAL 2-WIDE 2-INPUT AND-OR-INVERT GATES (ONE GATE EXPANDABLE)

51



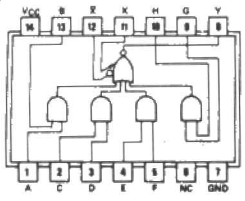
AND-OR-INVERT GATES

52



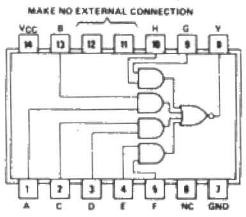
EXPANDABLE 4-WIDE AND-OR GATES

53



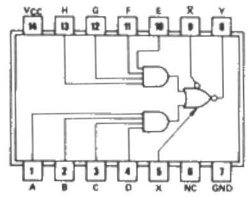
EXPANDABLE 4-WIDE AND-OR-INVERT GATES

54



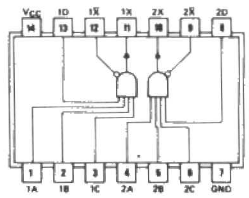
4-WIDE AND-OR-INVERT GATES

55



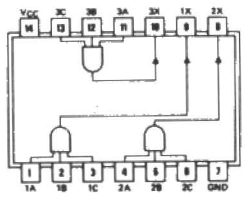
2-WIDE 4-INPUT AND-OR-INVERT GATES

60



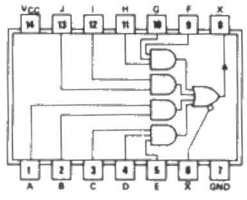
DUAL 4-INPUT EXPANDERS

61



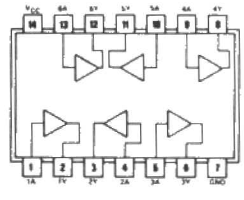
TRIPLE 3-INPUT EXPANDERS

62



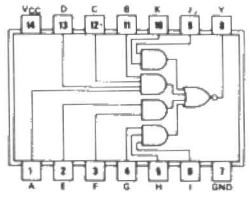
4-WIDE AND-OR EXPANDERS

63



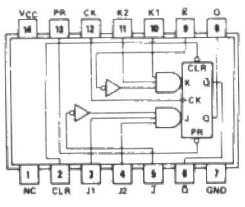
HEX CURRENT-SENSING INTERFACE GATES

64 65



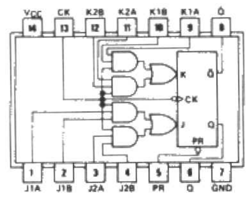
4-2-3-2 INPUT AND-OR-INVERT GATES

70



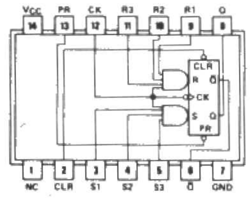
AND-GATED J-K POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

H71



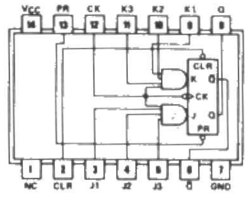
AND-OR-GATED J-K MASTER-SLAVE FLIP-FLOPS WITH PRESET JTPUTS

L71



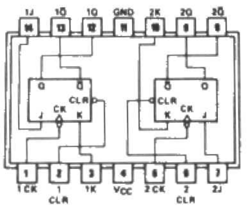
AND-GATED R-S MASTER-SLAVE FLIP-FLOPS WITH PRESET AND CLEAR

72



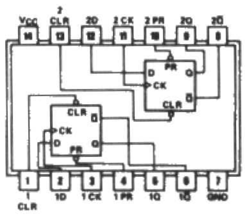
AND-GATED J-K MASTER-SLAVE FLIP-FLOPS WITH PRESET AND CLEAR

73



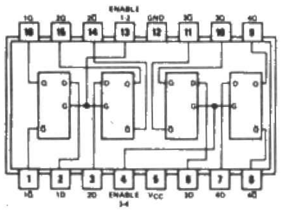
DUAL J-K FLIP-FLOPS WITH CLEAR

74



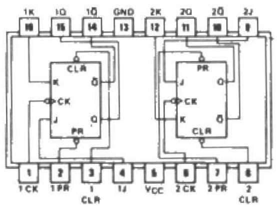
DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP -FLOPS WITH PRESET AND CLEAR

75


1Q 2Q 2Q̄ ENABLE 1-2 GND 3Q 3Q 4Q
D Q G Q̄
1Q̄ 1D 2D ENABLE 3-4 VCC 3D 4D 4Q̄

4-BIT BISTABLE LATCHES

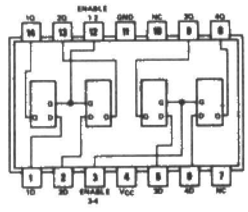
76


1K 1Q 1Q̄ GND 2K 2Q 2Q̄ 2J
CLR K CK J PR Q Q̄
1CK 1PR 1 CLR 1J VCC 2CK 2PR 2 CLR

DUAL J-K FLIP-FLOPS WITH PRESET AND CLEAR

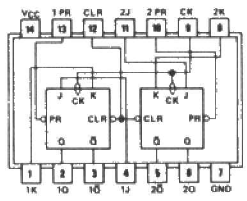
INTEGRATED DATA

77



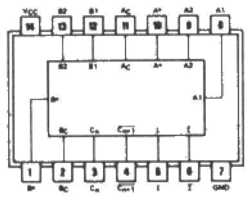
4-BIT BISTABLE LATCHES

78



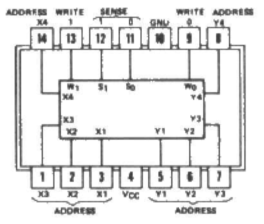
DUAL J-K
FLIP-FLOPS WITH PRESET, COMMON CLEAR, AND COMMON CLOCK

80



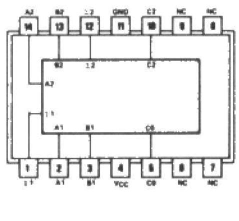
GATED FULL ADDERS

81



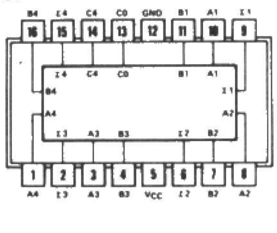
16-BIT RANDOM-ACCESS MEMORIES

82



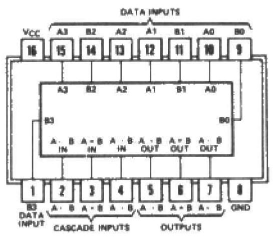
2-BIT BINARY FULL ADDERS ITS

83



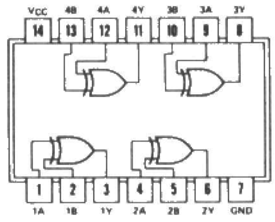
4-BIT BINARY
FULL ADDERS WITH FAST CARRY

85



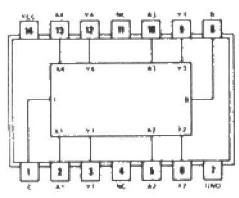
4-BIT MAGNITUDE COMPARATORS

86



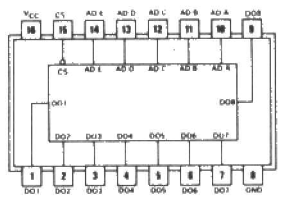
QUADRUPLE
2-INPUT EXCLUSIVE-OR GATES

87



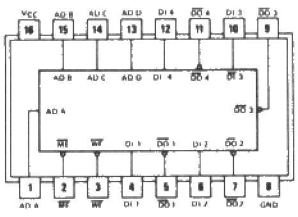
4-BIT TRUE/
COMPLEMENT, ZERO/ONE ELEMENTS

88



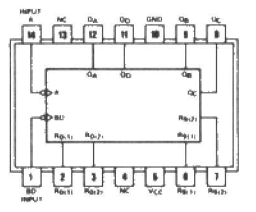
256-BIT READ-ONLY MEMORIES

89



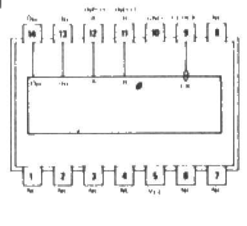
64-BIT READ/WRITE MEMORIES

90



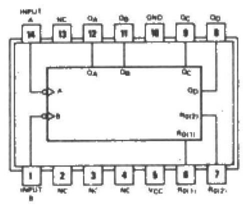
DECADE COUNTERS

91



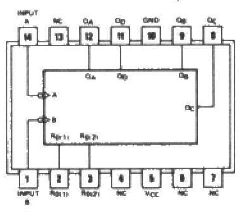
8-BIT SHIFT REGISTERS

92



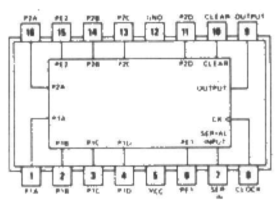
DIVIDE-BY-TWELVE COUNTERS

93



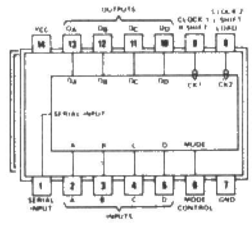
4-BIT BINARY COUNTERS PUTS

94



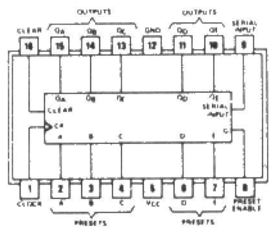
4-BIT SHIFT REGISTERS

95



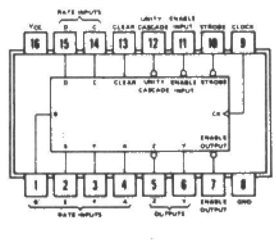
4-BIT SHIFT REGISTERS

96



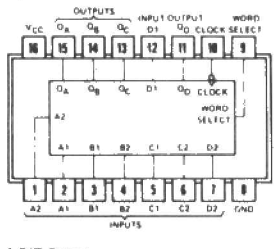
5-BIT SHIFT REGISTERS

97



SYNCHRONOUS 6-BIT BINARY
RATE MULTIPLIERS

98



4-BIT DATA
SELECTOR/STORAGE REGISTERS

"THERE MUST BE A COMPUTER DEALER I CAN TURN TO FOR GUIDANCE....."

If you're still staggering through the computer jungle and not getting sensible answers to your questions, we have some good news:

Now you can turn to **professional** people who are capable of giving you sound advice on practically all aspects of popular computing.

They all have one thing in common:

They are **COMPUTERS FOR ALL** dealers.

A Computers for All dealer is different from the normal Computer retailer.

Not surprisingly he won't try to sell you things like cameras or cosmetic stationery or sealing wax.

He will, however, be capable of answering sensibly almost any questio you have on computers and computing, and have readily available a wide range of popular computers, hardware, software, books, and peripherals.

So why not call in at your local **COMPUTERS FOR ALL** dealer today?

He can lead you in the direction you want to travel.

The shops where people matter:

AVON
JADE COMPUTERS
Coombend, Radstock, Bath. 0761 32570.
MERCATOR COMPUTER SYSTEMS
3 Whiteladies Road, Clifton, Bristol. 0272 731079.
MOBILE MICROS
2 Castle Street, Thornbury, Bristol. 0454 418383.

BERKSHIRE
KENNETH WARD COMPUTERS
Verve House, London Road, Sunningdale. 0990 25025.

CHANNEL ISLANDS
MEGA LTD.
7 Anley Street, St. Helier, Jersey. 0534 72263.

CHESHIRE
THE COMPUTER CENTRE
68 Chestergate, Macclesfield.

CORNWALL
COMPUTAVISION
4 Market Street, St. Austell. 0726 5900.
FAL-SOFT COMPUTERS
8 St. George's Arcade, Falmouth TR11 3DH. 0326 314663.

DEVON
A & D COMPUTERS
"Computerland", 6 City Arcade, Fore Street, Exeter. 0392 77117.
BITS & BYTES
44 Fore Street, Ilfracombe, N. Devon EX34 9JD. 0271 62801.
COMPUTER SYSTEMS (TORBAY)
Pump Street, Brixham. 08045 6565/6.
CRYSTAL COMPUTERS
209 Union Street, Torquay. 0803 22699.
SYNTAX LTD
Midhurst, Grenofen, Tavistock. 0822 2392.

DORSET
DENSHAM COMPUTERS
329 Ashley Road, Parkstone, Poole.

ESSEX
AKHTER INSTRUMENTS
Unit 19, Arlinghyde Estates, South Road, Harlow. 0279 412639.
COMPUTERS FOR ALL
72 North Street, Romford. 0708 752862.

HANTS
MICRO VIDEO STUDIOS LTD.
17 Turks Street, Alton. 0420 82055.

HERTS
VIDEO CITY
45-47 Fishers Green Road, Stevenage. 0438 53808.
THE COMBINED TRADING CO.
10 & 11 Salisbury Square, Old Hatfield. 07072 65551.
THE ROYSTON MICROCENTRE
1 John Street, Royston. 0763 42622.

KENT
APHROS SOFTWARE CO.
47 Hawley Square, Margate, Kent. 0843 294699.
MEDWAY COMPUTERS
141 New Road, Chatham. 0634 826080.
MICRO MAGIC
128 Erith Road, Bexleyheath. 0322 523052.
THE DATA STORE
43 Shepperton Road, Petts Wood. 0689 26698.
THE MICRO SHOP
11 The Pantiles, Tunbridge Wells. 0892 27691

LANCS
P.C.S.
39 Railway Road, Darwen. 0254 776677.
4 MAT COMPUTING
67 Friargate, Preston.
THE HOME COMPUTER CENTRE
40 King Street, Blackburn. 0254 671316.

LEICS
DIMENSION
27-29 High Street, Leicester. 0533 57479.

LONDON
KAYDE HOME COMPUTERS
1 Station Approach, New Eltham, London SE9. 01-859 7505.
KELLY'S COMPUTERMARKET
227 Dartmouth Road, Sydenham, London SE26 4QY. 01-699 4399/6202.

MIDDLESEX
SCREENS MICROCOMPUTERS
6 Main Avenue, Moor Park, Northwood. 09274 20664.
TWILLSTAR COMPUTERS
17 Regina Road, Southall. 01-574 5271.

N. IRELAND
D. V. MARTIN LTD.
13 Bridge Street, Belfast, BT1 1LT. 0232 226434.
MCLAUGHLIN ELECTRONICS
44 Carlisle Road, Londonderry. 0504 65002.

NORTHAMPTONSHIRE
LEISURETIME
13 The Friary, Grosvenor Centre, Northampton. 0604 36726.

OXFORDSHIRE
SCIENCE STUDIO
7 Little Clarendon, Oxford OX1 2HP. 0865 54022.

SHETLANDS
TOMORROW'S WORLD
Esplanade, Lerwick, ZE1 0LL. 0595 2145.

SUFFOLK
BECCLES COMPUTERS
5 The Ridings, Worlingham, Beccles. 0502 715061.

SURREY
ANIROG COMPUTERS
8 High Street, Horley. 02934 2007.
COMPUTASOLVE
8 Central Parade, St. Marks Hill, Surbiton. 01-390 5135.

SUSSEX
THE COMPUTER CENTRE (BMS) LTD.
37d & 37e Robertson Street, Hastings, East Sussex. 0424 439190.

S. WALES
AUTOMATION SERVICES (S. WALES)
3 Wermeys Road, Penysfai, Bridgend. 0656 720959.
DAN EVANS (BARRY) LTD.
81 Holton Road, Barry, South Glamorgan. 0446 734242.
MORRISTON COMPUTER CENTRE
37 Clase Road, Morriston, Swansea SA6 8DS. 0792 797572.
STEVE'S COMPUTER CO. LTD.
Castle Arcade, Cardiff, South Glamorgan. 0222 4 1905.

TYNE & WEAR
THE COMPUSHOP
10 Newgate Centre, Newcastle-upon-Tyne NE1 5RE. 0632 618673.

WARWICKSHIRE
IMPULSE MICRO SYSTEMS LTD.
6 Central Chambers, Cooks Alley, Wood Street, Stratford-Upon-Avon. 0789 295819.

WEST MIDLANDS
CALISTO COMPUTERS LTD.
119 John Bright Street, Birmingham B1 1BE. 021-632 6458.
JBC
200 Earlsdon Ave. North, Earlsdon, Coventry. 0203 73813.

WORCESTERSHIRE
DEATH VALLEY COMPUTERS
P.O. Box 54, Worcester WR2 6QA. 0905 640400.
EVESHAM COMPUTER CENTRE
Crown Court Yard, Bridge St., Evesham. 0386 48635.

YORKSHIRE
COM-TEC
6 Eastgate, Barnsley, South Yorkshire. 0226 46972.

COMPUTERS FOR ALL